```
December 9, 1978 5:58PM in <VANMELLE>WW.SAV;81121 by RGSMITH


Fns on CNET:

     !TOSPACE                                   PARSE!TASK!ABSTRACTION
     $INITIALIZE                                PROCESS!ACKNOWLEDGEMENT
     $TEST                                      PROCESS!ANNOUNCED!AWARD
     ADDATTRIBUTE                               PROCESS!BID
     ADDOBJECT                                  PROCESS!CONTRACT
     ANNOUNCE!TASK                              PROCESS!DIRECTED!AWARD
     ATTRIBUTEP                                 PROCESS!DISPLAY!EVENT
     AWARD                                      PROCESS!FINAL!REPORT
     BID                                        PROCESS!INFORMATION
     CHECK!BIDS                                 PROCESS!INTERIM!REPORT
     CHECK!ELIGIBILITY                          PROCESS!INTERNAL!EVENT
     CILP                                       PROCESS!MESSAGE
     CILPARSE                                   PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT
     CNET                                       PROCESS!REQUEST
     CNET*                                      PROCESS!TASK!ANNOUNCEMENT
     DEFINE!OBJECT                              PROCESS!TERMINATION
     DELETE!OBJECT                              QANNOUNCE
     DELETE!PSEUDO!CONTRACT                     QARANK
     DIRECTED!AWARD                             QBRANK
     DISPLAY!CONTRACT                           QDISPLAY
     DISPLAY!EVENT                              QFINALIZE
     DISPLAY!EVENTS!AT!TIME                     QINITIALIZE
     DISPLAY!MESSAGE                            QRECEIVE
     DISPLAY!NODE                               QSET!PARAMETERS
     DISPLAY!PARAMETERS                         RANDOMCOMPARE
     DISPLAY!RECORDS                            READY!CONTRACT
     DISPLAY!STATISTICS                         READYCOMPARE
     EXTEND!BOARD                               REANNOUNCE!TASK
     FINAL!REPORT                               RELEASE!TASK
     FIND!SUBCONTRACT                           RESIMULATE
     GENERATE!SUBTASK                           RESUME!TASK
     GET!TASK!ANNOUNCEMENT                      RETRIEVE!OBJECT
     GOOD!BOARD                                 SAME!STATUS!CHECK
     INITCIL                                    SDISPLAY
     INITIALIZE                                 SENDMESSAGE
     INSTALL!DISPLAY!EVENT                      SET!PARAMETERS
     INSTALL!EVENT                              SIMULATE
     INSTALL!INTERNAL!EVENT                     STORE!OBJECT
     INTERIM!REPORT                             STORE!TASK!OBJECT
     MAKE!BID                                   SUSPEND
     NEW!BOARD                                  TERMINATE
     NEXT!CONTRACT                              TERMINATE!SUBCONTRACTS
     NEXT!EVENT                                 UPDATE!ACTIVE!TASK!ANNOUNCEMENTS
     NODE!SEARCH                                UPDATE!NODE
     OBJECTP                                    UPDATE!OBJECT
     OUTSTANDING!SUBCONTRACTS                   UPDATE!TASK!TIME
     PARSE!NODE!ABSTRACTION                     VALUEP
```

```
rgs: 11-Oct-78 21:06 [CNET]                                                                            !TOSPACE
                                                                                                    ------------
(!TOSPACE
  [LAMBDA (x)                                                              (* rgs: "11-Oct-78 21:06")
    (MKSTRING (PACK (SUBST " " '! (UNPACK x])
----------
Called by: DISPLAY!EVENT DISPLAY!MESSAGE
```

Explanation: Replaces "!" with " " in atom names for cleaner output.

```
rgs: 10-Sep-78 08:46 [CNET]                                                                        $INITIALIZE
                                                                                                ---------------
($INITIALIZE
  [LAMBDA (xnetsize restartflag)                                           (* rgs: "10-Sep-78 08:46")
    (PROG (xprocedure xtask!template)
          (SETQ xprocedure (create PROCEDURE NAME ←'$TEST
                                   CODE ←'$TEST))
          (SETQ xtask!template (create TASK!TEMPLATE TYPE ←'$TEST
                                       EXECUTION!PROCEDURE ←'$TEST))
          (for x from 1 to 2 do (STORE!OBJECT x 'PROCEDURE xprocedure)
                                (STORE!OBJECT x 'TASK!TEMPLATE (COPYALL xtask!template)))
          (RETURN (LIST (LIST '$TEST "This is T1")
                        (LIST '$TEST "This is T2"])
----------
Calls:     STORE!OBJECT
```

Explanation: A sample initial applications function. Such a function is called to initialize nodes in the net with applications-specific information for the simulation. The arguments are "xnetsize", the number of processor nodes in the distributed architecture, "restartflag", a flag that is T if at least one simulation has already been performed, and "olduserparamflag", a flag that can be set to T by the user during interaction with CNET if the current user parameters are to be used as defaults during acquisition of new parameters. All I/O should be done directly and required CNET functions should be used directly without going through CNET* since this function is handled in a special manner, and not through the generator structure. The initial applications function returns a list of two-element lists of the form "(type specification)", where "type" is the type of task, and "specification" is the task specification. The returned N tasks in the list are assigned as top-level contracts to the first N processor nodes. See QINITIALIZE as an example for the N Queens problem.

```
rgs: 11-Oct-78 21:04 [CNET]                                                                              $TEST
                                                                                                     ---------
($TEST
  [LAMBDA (xpnode xname xspecification xcontract)                          (* rgs: "11-Oct-78 21:04")
    (PROG NIL
          (UPDATE!TASK!TIME 1)
          (CNET* 'SDISPLAY (LIST xspecification))
          (UPDATE!TASK!TIME 1)
          (CNET* 'GENERATE!SUBTASK (LIST (LIST '$TEST "This is subtask 1")))
          (UPDATE!TASK!TIME 1)
          (CNET* 'GENERATE!SUBTASK (LIST (LIST '$TEST "This is subtask 2")))
          (TERMINATE])
----------
Calls:     CNET* TERMINATE UPDATE!TASK!TIME
```

Explanation: A sample task execution procedure. Such a function is called to execute a task. The arguments are "xpnode", the name of the node in which the task is being executed, "xname", the name of the contract for the task, "xspecification", the task specification, and "xcontract", the complete contract record.
Such functions are implemented as generators and must access all CNET functions through CNET* (they are suspended each time a call to CNET* is made--for quasi parallelism).
Two special functions are available, SUSPEND, which moves the contract to the suspended state, and TERMINATE, which moves the contract to the terminated state.
No value is returned. See CNET*, SUSPEND, and TERMINATE for details on how they are called. See EXTEND!BOARD as an example for the N Queens problem.

```
rgs: 18-Aug-78 15:06 [CNET]                                                              ADDATTRIBUTE
                                                                                         ------------

(ADDATTRIBUTE
  [LAMBDA N                                                         (* rgs: "18-Aug-78 15:06")
    (for I from 1 to N do (COND
                           ([NOT (MEMBER (U-CASE (ARG N I))
                                         (GETPROP '#ATTRIBUTE 'POSSIBLEVALUES]
                             (ADDPROP '#ATTRIBUTE 'POSSIBLEVALUES (U-CASE (ARG N I])
----------
```

Called by: DEFINE!OBJECT

Explanation: Installs all of its arguments as 'attributes' in the common internode language. All attributes are first converted to upper case. Checks are made for duplication.

```
rgs: 18-Aug-78 15:04 [CNET]                                                                 ADDOBJECT
                                                                                            ---------

(ADDOBJECT
  [LAMBDA N                                                         (* rgs: "18-Aug-78 15:04")
    (for I from 1 to N do (COND
                           ([NOT (MEMBER (U-CASE (ARG N I))
                                         (GETPROP '#OBJECT 'POSSIBLEVALUES]
                             (ADDPROP '#OBJECT 'POSSIBLEVALUES (U-CASE (ARG N I])
----------
```

Called by: DEFINE!OBJECT

Explanation: Installs all of its arguments as 'objects' in the common internode language. All objects are first converted to upper case. Checks are made for duplication.

```
rgs: 16-Oct-78 21:20 [CNET]                                                             ANNOUNCE!TASK
                                                                                        -------------

(ANNOUNCE!TASK
  [LAMBDA (xpnode xname)                                            (* rgs: "16-Oct-78 21:20")
    (PROG (temp)
          (SETQ temp (GET!TASK!ANNOUNCEMENT xpnode xname))
          (COND
            [(EQUAL (CAR temp)
                    'DIRECTED!AWARD)
              (DIRECTED!AWARD xpnode xname (CADR temp)
                              (CADDR temp)
                              (CADDDR temp)
                              (CAR (CDDDDR temp]
            (T [SENDMESSAGE xpnode (IPLUS time ta)
                            (CAR temp)
                            (create TASK!ANNOUNCEMENT NAME ← xname ELIGIBILITY!SPECIFICATION ←(CADR temp)
                                    TASK!ABSTRACTION ←(CADDR temp)
                                    BID!SPECIFICATION ←(CADDDR temp)
                                    EXPIRATION!TIME ←(CAR (CDDDDR temp]
               (INSTALL!INTERNAL!EVENT (IPLUS time (CAR (CDDDDR temp)))
                                       xpnode xname 'BID!CHECK])
----------
```

Calls: DIRECTED!AWARD GET!TASK!ANNOUNCEMENT INSTALL!INTERNAL!EVENT SENDMESSAGE

Called by: GENERATE!SUBTASK REANNOUNCE!TASK

Freevars: ta time

Explanation: Sends either a directed award or a task announcement message for the contract with name "xname" that has been generated by node "xpnode". It also places a "bid!check" event on the event list to check for bids at the end of the expiration time. Uses GET!TASK!ANNOUNCEMENT to generate the task-dependent information for the directed award or task announcement.

```
rgs: 17-Oct-78 00:15 [CNET]                                                                    ATTRIBUTEP
                                                                                             --------------
(ATTRIBUTEP
  [LAMBDA (xobject xattribute)                                                 (* rgs: "17-Oct-78 00:15")
    (COND
      ((OBJECTP xobject)
        (COND
          ((MEMBER xattribute (RECORDFIELDNAMES (RECLOOK xobject)))
            T)
          (T (WRITE "CIL error: " xattribute " is not a valid attribute of " xobject)
             NIL])
----------
Calls:     OBJECTP
```

Explanation: Returns T if "xattribute" is a valid attribute of the object "xobject". If "xobject" is not a valid object or "xattribute" is not a valid attribute of "xobject" then WRITEs an error message and returns NIL.

```
rgs: 18-Sep-78 13:30 [CNET]                                                                        AWARD
                                                                                                 ---------
(AWARD
  [LAMBDA (xpnode xname xaddressee)                                            (* rgs: "18-Sep-78 13:30")
    (PROG (sc)
          (SETQ sc (NODE!SEARCH xpnode xname 'ANNOUNCED T T))
          [SENDMESSAGE xpnode (IPLUS time tpb tsaw)
                       xaddressee
                       (create ANNOUNCED!AWARD NAME ← xname TASK!SPECIFICATION ←(fetch (TASK SPECIFICATION)
                                                                                  of (RETRIEVE!OBJECT
                                                                                       xpnode
                                                                                       'TASK
                                                                                       (fetch (SUBCONTRACT TASK)
                                                                                          of (CAR sc]
                                                                               (* note the name of the contractor in
                                                                               the subcontract record)
          (replace (SUBCONTRACT CONTRACTOR) of (CAR sc) with xaddressee])
----------
Calls:     NODE!SEARCH RETRIEVE!OBJECT SENDMESSAGE

Called by: CHECK!BIDS PROCESS!BID

Freevars:  time tpb tsaw
```

Explanation: Sends an award message to "xaddressee" from "xpnode" for contract "xname". Updates the appropriate subcontract record.

rgs: 12-Sep-78 01:06 [CNET]   BID

```
(BID
  [LAMBDA (xpnode xcontract xmanager xtype xbid!specification)                 (* rgs: "12-Sep-78 01:06")
    (PROG (xpnode! xbidconsproc xnode!abstraction)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xbidconsproc (fetch (TASK!TEMPLATE BID!CONSTRUCTION!PROCEDURE) of (RETRIEVE!OBJECT xpnode
                                                                                                   'TASK!TEMPLATE xtype)
                                  ))
          [COND
            (xbidconsproc (SETQ xnode!abstraction (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                                     'PROCEDURE
                                                                                                     xbidconsproc))
                                                          (LIST xpnode xbid!specification]
          (SENDMESSAGE xpnode (IPLUS time tb)
                       xmanager
                       (create BID NAME ← xcontract NODE!ABSTRACTION ← xnode!abstraction])
----------
```

Calls: RETRIEVE!OBJECT SENDMESSAGE

Called by: MAKE!BID

Freevars: NET tb time

Explanation: Makes a bid on contract with name "xcontract" to manager "xmanager" from node "xpnode". The bid specification "xbid!specification" and the task type "xtype" are used to access the appropriate bid construction procedure and construct the bid.

rgs: 27-Oct-78 10:50 [CNET]   CHECK!BIDS

```
(CHECK!BIDS
  [LAMBDA (xpnode xname)                                                       (* rgs: "27-Oct-78 10:50")
    (PROG (xpnode! sc active!bids xawproc)
          (SETQ sc (NODE!SEARCH xpnode xname 'ANNOUNCED NIL T))
          (COND
            (sc (SETQ active!bids (CADR sc))
                [SETQ xawproc (fetch (TASK AWARD!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK (fetch (SUBCONTRACT TASK)
                                                                                                   of (CAR sc]
                [COND
                  (xawproc

          (* if there is an award procedure for the task then apply it to the list of bids
          (or the empty list if there are no bids))


                          (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xawproc))
                                 (LIST xpnode xname active!bids)))
                  (T                                                           (* otherwise if there are bids then make
                                                                               an announced award to the first one on
                                                                               the list -
                                                                               if no bids then reannounce)
                     (COND
                       [active!bids (AWARD xpnode xname (fetch (ACTIVE!BID CONTRACTOR) of (CAR active!bids]
                       (T (REANNOUNCE!TASK xpnode xname]
                (RETURN T))
            (T (RETURN NIL])
----------
```

Calls: AWARD NODE!SEARCH REANNOUNCE!TASK RETRIEVE!OBJECT

Called by: PROCESS!INTERNAL!EVENT

Explanation: Checks the bids on the contract with name "xname" in node "xpnode" at the end of the expiration time. If the contract has been awarded then returns T. If the contract has not been awarded then calls the award procedure for the task. If no award procedure exists, then awards the contract to the first bidder in the active!bids list. If no bids have been received, then reannounces the contract.

```
rgs: 27-Oct-78 10:23 [CNET]                                                          CHECK!ELIGIBILITY
                                                                                     -----------------

(CHECK!ELIGIBILITY
  [LAMBDA (xpnode elspec)                                             (* rgs: "27-Oct-78 10:23")
    (PROG (pelspec)
          (SETQ pelspec (for x in elspec always (CILPARSE x ELSPECGRAMMAR)))
          (RETURN pelspec])
----------
Calls:     CILPARSE

Called by: MAKE!BID PROCESS!DIRECTED!AWARD PROCESS!TASK!ANNOUNCEMENT

Freevars:  ELSPECGRAMMAR

Explanation:  Checks to see if node "xpnode" meets the eligibility specification "elspec", and, if so returns T.
           ANDS a series of statements written according to "elspecgrammar".



rgs: 17-Oct-78 22:04 [CNET]                                                                       CILP
                                                                                                  ----

(CILP
  [LAMBDA (word)                                                      (* rgs: "17-Oct-78 22:04")
    (COND
      ([SOME CILCLASSES (FUNCTION (LAMBDA (class)
                 (FMEMB word (GETPROP class 'POSSIBLEVALUES]
        T)
      (T NIL])
----------
Freevars:  CILCLASSES
```

```
(CILPARSE
  [LAMBDA (phrase grammar)                                    (* rgs: "25-Oct-78 01:21")
    (PROG (match)
          (SETQ match (INTERPRET phrase (COND
                                          (grammar grammar)
                                          (T CILGRAMMAR))
                                        CILCLASSES NIL T))
          (COND
            ((AND (fetch (INTERPRETATION MATCH) of match)
                  (NOT (fetch (INTERPRETATION REMAININGPHRASE) of match)))
              (RETURN match))
            ((fetch (INTERPRETATION MATCH) of match)
              (WRITE "Parse succeeded, but words remain in phrase")
              (WRITE)
              (WRITE "Results: " (fetch (INTERPRETATION RESULTS) of match))
              (WRITE "Remaining words: " (fetch (INTERPRETATION REMAININGPHRASE) of match))
              (WRITE "Bindings: " (fetch (INTERPRETATION BOUNDCLASSES) of match))
              (RETURN match))
            (T [COND
                 [(fetch (INTERPRETATION RESULTS) of match)          (* remove duplicate templates)
                   (replace (INTERPRETATION RESULTS) of match with (INTERSECTION (fetch (INTERPRETATION RESULTS)
                                                                                         of match)
                                                                                  (fetch (INTERPRETATION RESULTS)
                                                                                         of match)))
                   (WRITE "Parse failed: " (LENGTH (fetch (INTERPRETATION RESULTS) of match))
                          " template(s) partially matched")
                   (for x in (fetch (INTERPRETATION RESULTS) of match)
                      do (WRITE)
                         (WRITE '(Partially matched template:)
                                " "
                                (fetch (FAILURE TEMPLATE) of x))
                         (WRITE '(Semantic Predicate and Action Function:)
                                " "
                                (fetch (FAILURE FUNCTIONS) of x))
                         (WRITE '(Unmatched portion of template:)
                                " "
                                (fetch (FAILURE REMTEMPLATE) of x))
                         (WRITE '(Unmatched portion of phrase:)
                                " "
                                (fetch (FAILURE REMPHRASE) of x))
                         (WRITE 'Bindings:)
                         (for y in (fetch (FAILURE FBINDINGS) of x) do (WRITE "           " y))
                         (COND
                           ((NOT (fetch (FAILURE REMPHRASE) of x))
                             (WRITE "Semantic Predicate Failed"]
                 (T (WRITE '(Parse failed: No templates matched]
               (RETURN NIL])
----------
```

Called by: CHECK!ELIGIBILITY

Freevars: CILCLASSES CILGRAMMAR

Explanation: Parses "phrase" using CILGRAMMAR and CILCLASSES.

rgs: 26-Sep-78 01:13 [CNET]   **CNET**

```
(CNET
  [LAMBDA (restartflag oldcnetparamflag olduserparamflag)                    (* rgs: "26-Sep-78 01:13")
    (PROG (temp)
          (TTYOUT "                -----  CONTRACT NET Simulation -----")
          (TERPRI)
          (TERPRI)
          [COND
            ((NOT restartflag)
              (SET!PARAMETERS (NOT oldcnetparamflag]
          (SIMULATE restartflag olduserparamflag)
          (while (IGREATERP (SETQ temp (RESIMULATE))
                            0)
             do (COND
                  ((EQ temp 1)
                    (SIMULATE T T))
                  (T (SET!PARAMETERS)
                     (SIMULATE NIL T])
----------
```

Calls: RESIMULATE SET!PARAMETERS SIMULATE

Explanation: The top-level function in the CNET system. Starts a contract net simulation. "restartflag" is T if new parameters are not to be requested. "oldcnetparamflag" is T if the current cnet parameters are to be used as defaults when new cnet parameter are requested. "olduserparamflag" is T if the current user parameters are to be used as defaults when new user parameters are requested.

rgs: 7-Sep-78 05:33 [CNET]   **CNET***

```
(CNET*
  [LAMBDA (xfunction xarguments)                                             (* rgs: " 7-Sep-78 05:33")

          (* contract net system call -
          used by a user program to access contract net system functions -
          a user task processor (which is implemented as a generator through the possibilities list construct)
          is suspended when such a call is made -
          to give quasi-parallelism)


    (PROG NIL
          (COND
            ((EQ 'RELEASE (AU-REVOIR NIL))
              (ADIEU)                                                        (* used to release the generator when a
                                                                             contract has been terminated by the
                                                                             manager)
              ))
          (APPLY xfunction (APPEND (LIST xpnode xname)
                                   xarguments])
----------
```

Called by: $TEST EXTEND!BOARD QRECEIVE

Freevars: xname xpnode

Explanation: Used by a user function to access CNET functions. This is the only mode of access to CNET functions that is to be used by the user functions that actually execute tasks. Every time CNET* is called, the calling function is suspended so as to simulate parallelism. "xfunction" is the name of the CNET function to be applied. "xarguments" is the list of arguments for the function.

```
rgs: 16-Oct-78 21:54 [CNET]                                                                 DEFINE!OBJECT
                                                                                            -----------------
(DEFINE!OBJECT
  [LAMBDA N                                                                 (* rgs: "16-Oct-78 21:54")
    (PROG (temp)
          [SETQ temp (CONS 'TYPERECORD (CONS (U-CASE (ARG N 1))
                                             (LIST (for I from 2 to N collect (U-CASE (ARG N I]
          (EVAL temp)
          (ADDOBJECT (ARG N 1))
          (APPLY 'ADDATTRIBUTE (for I from 2 to N collect (ARG N I])
----------
Calls:     ADDATTRIBUTE ADDOBJECT
```

Explanation: Makes a TYPERECORD declaration using the first argument as the type of record. Calls ADDOBJECT and ADDATTRIBUTE to add the type of record to the list of objects defined in the common internode language, and the rest of the arguments to the list of attributes.

```
rgs:  8-Sep-78 03:11 [CNET]                                                                 DELETE!OBJECT
                                                                                            -----------------
(DELETE!OBJECT
  [LAMBDA (xpnode xobject xkey)                                             (* rgs: " 8-Sep-78 03:11")
    (PROG (xpnode! kb index otherindex otherindex1 xinstance)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ kb (fetch (PNODE KNOWLEDGE!BASE) of xpnode!))
          [COND
            [(MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE!BASE))
              [SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE]
              [SETQ xinstance (CAR (SOME index (FUNCTION (LAMBDA (x)
                                                 (EQUAL (CADR x)
                                                        xkey]
              (COND
                (xinstance (SETQ index (REMOVE xinstance index))
                           (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE)
                                         'replace index]
            (T (SETQ index (fetch (KNOWLEDGE!BASE OTHER) of kb))
               [SETQ otherindex (CAR (SOME index (FUNCTION (LAMBDA (x)
                                                   (EQUAL (CAR x)
                                                          xobject]
               [SETQ xinstance (CAR (SOME (CDR otherindex)
                                          (FUNCTION (LAMBDA (x)
                                              (EQUAL (CADR x)
                                                     xkey]
               (COND
                 (xinstance (SETQ otherindex1 (REMOVE xinstance otherindex))
                            (SETQ index (SUBST otherindex1 otherindex index))
                            (replace (KNOWLEDGE!BASE OTHER) of kb with index]
          (RETURN xinstance])
----------
Called by: DELETE!PSEUDO!CONTRACT PROCESS!TERMINATION UPDATE!NODE

Freevars:  NET
```

Explanation: Removes an object from the knowledge base of node "xpnode". "xobject" is the 'type' of object, and "xkey" is the key that specifies the object. All objects are represented as record structures and the key must be the first field in the record structure for the object to be deleted.

edit: 18-Sep-78 06:13 [CNET]   DELETE!PSEUDO!CONTRACT

```
(DELETE!PSEUDO!CONTRACT
  [LAMBDA (xpnode xname)                                     (* edit: "18-Sep-78 06:13")
    (PROG NIL                                                (* a pseudo-contract has state "pseudo")
          (COND
            ((EQUAL (fetch (CONTRACT STATE) of (RETRIEVE!OBJECT xpnode 'CONTRACT xname))
                    'PSEUDO)
              (DELETE!OBJECT xpnode 'CONTRACT xname)
              (RETURN T))
            (T (RETURN])
```

Calls: DELETE!OBJECT RETRIEVE!OBJECT

Called by: PROCESS!INTERNAL!EVENT

Explanation: Removes the pseudo!contract with name "xname" from node "xpnode". If the contract was not awarded, then the pseudo-contract that was set up when a bid was made still has state 'pseudo'.

rgs: 27-Sep-78 21:36 [CNET]   DIRECTED!AWARD

```
(DIRECTED!AWARD
  [LAMBDA (xpnode xname xaddressee xes xta xts)              (* rgs: "27-Sep-78 21:36")
    (PROG (sc)
          (SETQ sc (CAR (NODE!SEARCH xpnode xname 'ANNOUNCED T)))
          (SENDMESSAGE xpnode (IPLUS time tdaw)
                       xaddressee
                       (create DIRECTED!AWARD NAME ← xname ELIGIBILITY!SPECIFICATION ← xes TASK!ABSTRACTION ← xta
                               TASK!SPECIFICATION ← xts))    (* note the name of the prospective
                                                             contractor in the subcontract record -
                                                             bound to a "*" to indicate that
                                                             acknowledgement has not yet been
                                                             received)
          (replace (SUBCONTRACT CONTRACTOR) of sc with (CONS xaddressee '*])
```

Calls: NODE!SEARCH SENDMESSAGE

Called by: ANNOUNCE!TASK

Freevars: tdaw time

Explanation: Sends a directed award message to "xaddressee" from "xpnode" for the contract with name "xname". Uses "xes" as eligibility specification, "xabs" as task abstraction, and "xts" as task specification.

rgs: 16-Jul-78 12:13 [CNET]   DISPLAY!CONTRACT

```
(DISPLAY!CONTRACT
  [LAMBDA (c)                                                (* rgs: "16-Jul-78 12:13")
    (PROG NIL
          (DISPLAY "+name: " (fetch (CONTRACT NAME) of c))
          (DISPLAY " manager: " (fetch (CONTRACT MANAGER) of c))
          (DISPLAY "  subcontract/subcontractor:" (for x in (CDR (fetch (CONTRACT SUBCONTRACTS) of c))
                                                  collect (LIST (fetch (SUBCONTRACT NAME) of x)
                                                                (fetch (SUBCONTRACT CONTRACTOR) of x])
```

Explanation: Displays the name, manager, subcontract names and subcontracts for the contract record "c".

```
(DISPLAY!EVENT
  [LAMBDA (e forceflag)                                                   (* edit: "18-Sep-78 06:14")
    (PROG NIL
          (SELECTQ (CAR (fetch (EVENT DATA) of e))
                   [DISPLAY!EVENT (COND
                                    ((OR forceflag (AND (EQ (fetch (DISPLAY!EVENT TYPE) of (fetch (EVENT DATA)
                                                                                                   of e))
                                                                 'SIMULATION)
                                                              display!display!events!flag))
                                                                          (* two types of display!event -
                                                                          task and simulation -
                                                                          task display!events are always processed
                                                                          whereas simulation display!events are
                                                                          only processed when
                                                                          display!display!events!flag is set)
                                      (PROCESS!DISPLAY!EVENT (fetch (EVENT DATA) of e]
                   [INTERNAL!EVENT (COND
                                     ((OR forceflag display!internal!events!flag)
                                       (DISPLAY)
                                       (DISPLAY "node: " (fetch (INTERNAL!EVENT PNODE) of (fetch (EVENT DATA)
                                                                                                 of e)))
                                       (DISPLAY "contract: " (fetch (INTERNAL!EVENT NAME) of (fetch (EVENT DATA)
                                                                                                   of e)))
                                       [DISPLAY "internal event: " (!TOSPACE (fetch (INTERNAL!EVENT TYPE)
                                                                                 of (fetch (EVENT DATA) of e]
                                       (DISPLAY]
                   [MESSAGE (COND
                              ((OR forceflag display!messages!flag)
                                (DISPLAY!MESSAGE (fetch (EVENT DATA) of e]
                   NIL])
----------
```

Calls: !TOSPACE DISPLAY!MESSAGE PROCESS!DISPLAY!EVENT

Called by: DISPLAY!EVENTS!AT!TIME SIMULATE

Freevars: display!display!events!flag display!internal!events!flag display!messages!flag

Explanation: Displays the particulars of the event "e". If "forceflag" is T then the normal display flags are overridden, and the event is always displayed.

```
rgs: 10-Sep-78 13:34 [CNET]                                                                      DISPLAY!EVENTS!AT!TIME
                                                                                                ---------------------------
(DISPLAY!EVENTS!AT!TIME
  [LAMBDA (t forceflag)                                                                  (* rgs: "10-Sep-78 13:34")
    (PROG (e)
          (SETQ e eventlist)
          (while e do (COND
                        [(ILESSP t (fetch (EVENT TIME) of e))
                          (COND
                            ((fetch (EVENT LLINK) of e)
                              (SETQ e (fetch (EVENT LLINK) of e)))
                            (T (GO $$OUT]
                        [(IGREATERP t (fetch (EVENT TIME) of e))
                          (COND
                            ((fetch (EVENT RLINK) of e)
                              (SETQ e (fetch (EVENT RLINK) of e)))
                            (T (GO $$OUT]
                        (T (while (EQ t (fetch (EVENT TIME) of e)) do (DISPLAY!EVENT e forceflag)
                                                                      (COND
                                                                        ((fetch (EVENT RLINK) of e)
                                                                          (SETQ e (fetch (EVENT RLINK) of e)))
                                                                        (T (SETQ e NIL)
                                                                           (GO $$OUT])
----------
Calls:     DISPLAY!EVENT

Freevars:  eventlist

Explanation:  Displays the particulars of all events scheduled for time "t". If "forceflag" is T then the normal
           display flags are overridden, and the events are always displayed.
```

```
rgs: 15-Aug-78 10:48 [CNET]                                                                          DISPLAY!MESSAGE
                                                                                                    --------------------
(DISPLAY!MESSAGE
  [LAMBDA (m)                                                                            (* rgs: "15-Aug-78 10:48")
    (PROG NIL
          (DISPLAY)
          (DISPLAY "To: " (fetch (MESSAGE ADDRESSEE) of m))
          (DISPLAY "From: " (fetch (MESSAGE ORIGINATOR) of m))
          [DISPLAY "Type: " (!TOSPACE (CAR (fetch (MESSAGE TEXT) of m]
          (DISPLAY "Contract: " (CADR (fetch (MESSAGE TEXT) of m)))
          (DISPLAY])
----------
Calls:     !TOSPACE

Called by: DISPLAY!EVENT

Explanation:  Displays the addressee, originator, type, and contract name for message "m".
```

```
(DISPLAY!NODE
  [LAMBDA (xpnode forceflag)                                                     (* rgs: "15-Sep-78 21:50")
    (PROG (xpnode!)
          (SETQ xpnode! (ELT NET xpnode))
          (COND
            ((OR forceflag (EQUAL (fetch (PNODE STATUS) of xpnode!)
                                  "Busy")
                 (fetch (PNODE ANNOUNCED) of xpnode!))
              (DISPLAY)
              (DISPLAY "Node " xpnode)
              [DISPLAY "Executing: " (LIST (fetch (CONTRACT NAME) of (CAR (fetch (PNODE EXECUTING) of xpnode!]
              [DISPLAY "Ready: " (for x in (fetch (PNODE READY) of xpnode!) collect (fetch (CONTRACT NAME)
                                                                                          of (CAR x]
              [DISPLAY "Announced: " (for x in (fetch (PNODE ANNOUNCED) of xpnode!) collect (fetch (SUBCONTRACT NAME)
                                                                                                  of (CAR x]
              [DISPLAY "Suspended: " (for x in (fetch (PNODE SUSPENDED) of xpnode!) collect (fetch (CONTRACT NAME)
                                                                                                  of (CAR x]
              (DISPLAY "Terminated: " (for x in (fetch (PNODE TERMINATED) of xpnode!) collect (fetch (CONTRACT NAME)
                                                                                                    of x)))
              (DISPLAY])
----------
```

Called by: SIMULATE

Freevars: NET

Explanation: Displays the names of the contracts and subcontracts in the contract processing states of node "xpnode". If "forceflag" is T then the names are always displayed. Otherwise, they are only displayed if the node is "Busy".

```
(DISPLAY!PARAMETERS
  [LAMBDA NIL                                                            (* rgs: " 5-Sep-78 23:32")
    (DISPLAY)
    (DISPLAY "  CONTRACT NET Simulation Parameters
")
    (DISPLAY "     Number of Processor Nodes in Net: " netsize)
    (DISPLAY "     Applications time unit expansion: " gain)
    (DISPLAY "     Contracts held in terminated state: " ntermcs)
    (DISPLAY "
  CONTRACT NET Delay Parameters
")
    (DISPLAY "     Time to make a task announcement: " ta)
    (DISPLAY "     Time before a task is reannounced: " tra)
    (DISPLAY "     Time to process a task announcement: " tpa)
    (DISPLAY "     Time to make a node availability announcement: " tna)
    (DISPLAY "     Time to process a node availability announcement: " tpna)
    (DISPLAY "     Time to make a bid: " tb)
    (DISPLAY "     Time to process a bid: " tpb)
    (DISPLAY "     Time to make an announced award: " tsaw)
    (DISPLAY "     Time to process an announced award: " tpsaw)
    (DISPLAY "     Time to make a directed award: " tdaw)
    (DISPLAY "     Time to process a directed award: " tpdaw)
    (DISPLAY "     Time to acknowledge a directed award: " tack)
    (DISPLAY "     Time to process an acknowledgement: " tpack)
    (DISPLAY "     Time to make a report: " tr2)
    (DISPLAY "     Time to process a report: " tpr)
    (DISPLAY "     Time to generate a termination: " tt)
    (DISPLAY "     Time to process a termination: " tpt)
    (DISPLAY "     Time to generate a request: " treq)
    (DISPLAY "     Time to process a request: " tpreq)
    (DISPLAY "     Time to generate an information message: " ti)
    (DISPLAY "     Time to process an information message: " tpi])
----------
```

Called by: SIMULATE

Freevars: gain netsize ntermcs ta tack tb tdaw ti tna tpa tpack tpb tpdaw tpi tpna tpr tpreq tpsaw tpt tr2 tra treq tsaw tt

Explanation: Displays the CONTRACT NET simulation parameters.

```
(DISPLAY!RECORDS
  [LAMBDA NIL                                             (* rgs: "27-Oct-78 10:57")
    (PROG NIL                                             (* this is to get the record definitions
                                                          to show up in the LISTFNS file)
                                                          (* it must be updated if a record      |
                                                          definition is changed or added)
          '(TYPERECORD PNODE (UTILIZATION STATUS EXECUTING READY ANNOUNCED SUSPENDED TERMINATED
                                ACTIVE!TASK!ANNOUNCEMENTS KNOWLEDGE!BASE TASKCOUNTER))
          '(TYPERECORD KNOWLEDGE!BASE (CONTRACT TASK!TEMPLATE TASK NODE PROCEDURE DEVICE POSITION OTHER))
          '(TYPERECORD NODE (NAME DEVICE POSITION))
          '(TYPERECORD DEVICE (NAME TYPE NUMBER))
          '(TYPERECORD POSITION (NAME AREA LAT LONG))
          '(TYPERECORD CONTRACT (NAME MANAGER REPORT!RECIPIENTS RELATED!CONTRACTORS TASK RESULTS SUBCONTRACTS STATE))
          '(TYPERECORD SUBCONTRACT (NAME CONTRACTOR TASK RESULTS PREDECESSORS SUCCESSORS))
          '(TYPERECORD TASK!TEMPLATE (TYPE ANNOUNCEMENT!PROCEDURE ANNOUNCEMENT!RANKING!PROCEDURE
                                BID!CONSTRUCTION!PROCEDURE BID!RANKING!PROCEDURE AWARD!PROCEDURE
                                REFUSAL!PROCEDURE REFUSAL!PROCESSING!PROCEDURE REPORT!ACCEPTANCE!PROCEDURE
                                TERMINATION!PROCEDURE INFORMATION!ACCEPTANCE!PROCEDURE EXECUTION!PROCEDURE
                                TASKS))
          '(TYPERECORD TASK (NAME TYPE ANNOUNCEMENT!PROCEDURE ANNOUNCEMENT!RANKING!PROCEDURE BID!CONSTRUCTION!PROCEDURE
                         BID!RANKING!PROCEDURE AWARD!PROCEDURE REFUSAL!PROCEDURE REFUSAL!PROCESSING!PROCEDURE
                         REPORT!ACCEPTANCE!PROCEDURE TERMINATION!PROCEDURE INFORMATION!ACCEPTANCE!PROCEDURE
                         EXECUTION!PROCEDURE SPECIFICATION))
          '(TYPERECORD PROCEDURE (NAME CODE))
          '(TYPERECORD EVENT (TIME DATA LLINK RLINK))
          '(TYPERECORD INTERNAL!EVENT (PNODE NAME TYPE DATA))
          '(TYPERECORD DISPLAY!EVENT (PNODE TYPE DATA))
          '(TYPERECORD MESSAGE (TIME ADDRESSEE ORIGINATOR TEXT))
          '(TYPERECORD TASK!ANNOUNCEMENT (NAME ELIGIBILITY!SPECIFICATION TASK!ABSTRACTION BID!SPECIFICATION
                                    EXPIRATION!TIME))
          '(TYPERECORD ACTIVE!TASK!ANNOUNCEMENT (MANAGER CONTRACT TYPE ABSTRACTION BID!SPECIFICATION TIME
                                             EXPIRATION!TIME))
          '(TYPERECORD ACTIVE!BID (CONTRACTOR ABSTRACTION TIME))
          '(TYPERECORD NODE!AVAILABILITY!ANNOUNCEMENT (NODE!ABSTRACTION ELIGIBILITY!SPECIFICATION EXPIRATION!TIME))
          '(TYPERECORD BID (NAME NODE!ABSTRACTION))
          '(TYPERECORD ANNOUNCED!AWARD (NAME TASK!SPECIFICATION))
          '(TYPERECORD DIRECTED!AWARD (NAME ELIGIBILITY!SPECIFICATION TASK!ABSTRACTION TASK!SPECIFICATION))
          '(TYPERECORD ACKNOWLEDGEMENT (NAME RESPONSE REFUSAL!JUSTIFICATION))
          '(TYPERECORD INTERIM!REPORT (NAME RESULT!DESCRIPTION))
          '(TYPERECORD FINAL!REPORT (NAME RESULT!DESCRIPTION))
          '(TYPERECORD TERMINATION (NAME))
          '(TYPERECORD REQUEST (NAME REQUEST!SPECIFICATION))
          '(TYPERECORD INFORMATION (NAME INFORMATION!SPECIFICATION))
          '(TYPERECORD BOARD (COLUMN Q A B C QUEENS])
----------
Explanation:  Included so that CONTRACT NET record definitions appear in a file generated with "listfns".
```

```
(DISPLAY!STATISTICS
  [LAMBDA NIL                                                                   (* rgs: "25-Oct-78 02:11")
    (PROG (k ptu ptu2 temp)
          (DISPLAY)
          (DISPLAY "Time Units to Completion:" rtime)
          (DISPLAY)
          (DISPLAY "Communications Traffic Summary")
          (DISPLAY "------------------------------")
          (DISPLAY)
          (DISPLAY "Number of messages: " messagecounter)
          (DISPLAY "Number of broadcast messages: " bdcstcounter)
          (DISPLAY "Number of task announcements: " tacounter)
          (DISPLAY "Number of bids: " bidcounter)
          (DISPLAY "Number of announced awards: " aacounter)
          (DISPLAY "Number of directed awards: " dacounter)
          (DISPLAY "Number of acceptances: " acccounter)
          (DISPLAY "Number of refusals: " recounter)
          (DISPLAY "Number of interim reports: " ircounter)
          (DISPLAY "Number of final reports: " frcounter)
          (DISPLAY "Number of terminations: " tecounter)
          (DISPLAY "Number of node availability announcements: " nacounter)
          (DISPLAY "Number of requests: " rqcounter)
          (DISPLAY "Number of information messages: " imcounter)
          (DISPLAY)
          (DISPLAY "Number of events: " eventcounter)
          (DISPLAY)
          (DISPLAY "Number of task re-announcements: " tracounter)
          (DISPLAY)
          (DISPLAY "Processor Node Utilization Statistics")
          (DISPLAY "-------------------------------------")
          (DISPLAY)
          (DISPLAY "     Node        Utilization")
          (SETQ ptu 0)
          (SETQ ptu2 0)
          (SETQ k 0)
          [for xpnode from 1 to netsize do (COND
                                             ((IGREATERP (SETQ temp (IDIFFERENCE (ELT utilization xpnode)
                                                                                  1))
                                                         0)
                                               (SETQ k (ADD1 k))
                                               (SETQ ptu (IPLUS ptu temp))
                                               (SETQ ptu2 (IPLUS ptu2 (ITIMES temp temp)))
                                               (DISPLAY "       " xpnode "          " (FQUOTIENT temp rtime]
          (DISPLAY)
          (DISPLAY "Mean Processor Node Utilization: " (FQUOTIENT ptu (ITIMES k rtime)))
          (DISPLAY "Standard Deviation: " (FQUOTIENT (SQRT (FQUOTIENT (FDIFFERENCE ptu2 (FQUOTIENT (ITIMES ptu ptu)
                                                                                                      k))
                                                                      (SUB1 k)))
                                                     rtime])
```

Called by: SIMULATE

Freevars: aacounter acccounter bdcstcounter bidcounter dacounter eventcounter frcounter imcounter ircounter messagecounter nacounter netsize recounter rqcounter rtime tacounter tecounter tracounter utilization

Explanation: Displays processor node utilization statistics for the simulation.

```
(EXTEND!BOARD
  [LAMBDA (xpnode xname xspecification xcontract)                    (* rgs: "16-Oct-78 21:54")
    (PROG (xboard nextrow xrow subtaskflag solutions)
          (SETQ nextrow 1)
          (do (SETQ xboard (NEW!BOARD (fetch (BOARD COLUMN) of xspecification)
                                      (fetch (BOARD Q) of xspecification)
                                      (fetch (BOARD A) of xspecification)
                                      (fetch (BOARD B) of xspecification)
                                      (fetch (BOARD C) of xspecification)))
              (SETA (fetch (BOARD Q) of xboard)
                    (fetch (BOARD COLUMN) of xboard)
                    nextrow)                                         (* augment the time by the time it takes
                                                                     to generate a new board)
              (UPDATE!TASK!TIME tqgenerate)
              [COND
                ((GOOD!BOARD xboard)
                  (SETQ subtaskflag T)                               (* a valid board has been generated -
                                                                     start up another processor to extend it)
                                                                     (* first check to see if it is a
                                                                     solution to the problem -
                                                                     if so, then report success)
                  (COND
                    ((EQUAL (fetch (BOARD COLUMN) of xboard)
                            qsize)                                   (* augment the time by the time it takes
                                                                     to decide that a complete board has been
                                                                     generated)
                      (UPDATE!TASK!TIME tqsuccess)                   (* report success)
                      [CNET* 'SDISPLAY (LIST (CONS "Generated Board-->" (QDISPLAY (fetch (BOARD Q) of xboard]
                      [replace (CONTRACT RESULTS) of xcontract with (LIST (LIST 'SUCCESS 'BOARD 'Q (fetch (BOARD Q)
                                                                                                  of xboard]
                      [CNET* 'FINAL!REPORT (LIST (LIST (LIST 'SUCCESS 'BOARD 'Q (fetch (BOARD Q) of xboard]
                      (TERMINATE)))
                  (SETQ xrow (ELT (fetch (BOARD Q) of xboard)
                                  (fetch (BOARD COLUMN) of xboard)))
                  (SETA (fetch (BOARD A) of xboard)
                        xrow T)
                  (SETA (fetch (BOARD B) of xboard)
                        (IPLUS (fetch (BOARD COLUMN) of xboard)
                               xrow)
                        T)
                  (SETA (fetch (BOARD C) of xboard)
                        (IDIFFERENCE (IPLUS qsize xrow)
                                     (fetch (BOARD COLUMN) of xboard))
                        T)
                  (replace (BOARD COLUMN) of xboard with (ADD1 (fetch (BOARD COLUMN) of xboard)))
                                                                     (* augment the time by the time it takes
                                                                     to package a subtask)
                  (UPDATE!TASK!TIME tqsubtask)
                  [CNET* 'SDISPLAY (LIST (CONS "Generated Board-->" (QDISPLAY (fetch (BOARD Q) of xboard]
                  (CNET* 'GENERATE!SUBTASK (LIST (LIST 'EXTEND!BOARD xboard]
              (SETQ nextrow (ADD1 nextrow)) until (IGREATERP nextrow qsize))
          (COND
            [subtaskflag (COND
                           ((fetch (CONTRACT RESULTS) of xcontract)
                             (QRECEIVE xpnode xname xcontract))
                           (T (SUSPEND)
                              (QRECEIVE xpnode xname xcontract]
            (T                                                       (* augment the time by the time it takes
                                                                     to determine that no valid boards can be
                                                                     generated)
               (UPDATE!TASK!TIME tqfailure)                          (* report failure)
               [replace (CONTRACT RESULTS) of xcontract with '((FAILURE]
               [CNET* 'FINAL!REPORT (LIST (LIST (LIST 'FAILURE]
               (TERMINATE])
----------
Calls:    CNET* GOOD!BOARD NEW!BOARD QDISPLAY QRECEIVE SUSPEND TERMINATE UPDATE!TASK!TIME
```

Freevars: qsize tqfailure tqgenerate tqsubtask tqsuccess

Explanation: The task execution function for the N Queens 'extend!board' task. "xpnode" is the node in which the function is being executed. "xname" is the name of the contract. "xspecification" is the 'task!specification', and "xcontract" is the contract record.
   Generates subtasks by adding 1 new queen to the existing board to each possible row for the next column. Accepts reports and passes them on according to the report strategy set up in QSET!PARAMETERS. Updates task time for a realistic of concurrency.

edit: 18-Sep-78 07:36 [CNET]   FINAL!REPORT

```
(FINAL!REPORT
  [LAMBDA (xpnode xname xrslt xaddressee)                          (* edit: "18-Sep-78 07:36")
    (COND
      ((SAME!STATUS!CHECK xpnode xname)
        (PROG (xpnode! xcontract xreport)
              (SETQ xpnode! (ELT NET xpnode))
              (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT xname))
              (COND
                ((NOT xaddressee)
                  (SETQ xaddressee (fetch (CONTRACT REPORT!RECIPIENTS) of xcontract))
                                                                   (* default addressee is the list of
                                                                   report recipients for the contract)
                ))
              (SETQ xreport (create FINAL!REPORT NAME ← xname RESULT!DESCRIPTION ← xrslt))
              (SENDMESSAGE xpnode (IPLUS time tr2)
                           xaddressee xreport])
----------
```

Calls: RETRIEVE!OBJECT SAME!STATUS!CHECK SENDMESSAGE

Freevars: NET time tr2

Explanation: Sends a final report from "xpnode" to "xaddressee" for the contract with name "xname". If "xaddressee" is NIL, then the report is sent to the report!recipients for the contract. The text of the report is "xrslt".

rgs: 27-Sep-78 20:42 [CNET]   FIND!SUBCONTRACT

```
(FIND!SUBCONTRACT
  [LAMBDA (xpnode xname)                                           (* rgs: "27-Sep-78 20:42")
    (PROG NIL
          (RETURN (CAR (SOME [CDR (fetch (CONTRACT SUBCONTRACTS) of (RETRIEVE!OBJECT xpnode 'CONTRACT (CDR xname]
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL (fetch (SUBCONTRACT NAME) of x)
                                        xname])
----------
```

Calls: RETRIEVE!OBJECT

Called by: GENERATE!SUBTASK PROCESS!FINAL!REPORT TERMINATE!SUBCONTRACTS

Explanation: Returns the subcontract with name "xname" in the node "xpnode". The function searches the subcontracts slot of the contract with name (CDR xname).

```
(GENERATE!SUBTASK
  [LAMBDA (xpnode xname xsubtask xpredecessors)                          (* rgs: "27-Sep-78 23:48")
                                                                         (* predecessors is a list of the names
                                                                         of subtasks previously generated from
                                                                         the task that must be completed before
                                                                         the new subtask can be announced)
    (COND
      ((SAME!STATUS!CHECK xpnode xname)
        (PROG (xpnode! scl sc sname xsubtaskname)
              (SETQ xpnode! (ELT NET xpnode))                            (* update the number of subcontracts
                                                                         generated from the contract)
              [SETQ scl (fetch (CONTRACT SUBCONTRACTS) of (CAR (fetch (PNODE EXECUTING) of xpnode!]
              [COND
                [scl (FRPLACA scl (ADD1 (CAR scl]
                (T (SETQ scl (LIST 1]                                    (* create a subcontract structure for
                                                                         the new subtask)
              (SETQ xsubtaskname (STORE!TASK!OBJECT xpnode (CAR xsubtask)
                                                    (CADR xsubtask)))
              [COND
                (xpredecessors (SETQ xpredecessors (for x in xpredecessors collect x
                                                     when (PROG (tmpsc)
                                                                (SETQ tmpsc (FIND!SUBCONTRACT xpnode x))
                                                                (COND
                                                                  (tmpsc (replace (SUBCONTRACT SUCCESSORS) of tmpsc
                                                                                  with (CONS (fetch (SUBCONTRACT NAME)
                                                                                                    of sc)
                                                                                             (fetch (SUBCONTRACT SUCCESSORS)
                                                                                                    of tmpsc)))
                                                                         (RETURN T))
                                                                  (T (RETURN]
              (SETQ sc (create SUBCONTRACT NAME ←[CONS (CAR scl)
                                                      (fetch (CONTRACT NAME) of (CAR (fetch (PNODE EXECUTING)
                                                                                            of xpnode!]
                               CONTRACTOR ← 0 TASK ← xsubtaskname PREDECESSORS ← xpredecessors))
                                                                         (* place the new structure on the list
                                                                         of subcontracts for the contract)
              (FRPLACD scl (CONS sc (CDR scl)))
              (replace (CONTRACT SUBCONTRACTS) of (CAR (fetch (PNODE EXECUTING) of xpnode!)) with scl)

          (* find any outstanding (i.e., not yet completed) subcontracts in the announced state that
          correspond to members of the predecessors specified -
          mark the new subcontract as a successor -
          the actual outstanding subcontracts become the actual predecessors bound to the new subcontract in
          the announced state)

                                                                         (* place the new subcontract in the
                                                                         announced state, bound to the
                                                                         active!bids)
              (replace (PNODE ANNOUNCED) of xpnode! with (CONS (CONS sc NIL)
                                                               (fetch (PNODE ANNOUNCED) of xpnode!)))
                                                                         (* if there are no predecessors yet
                                                                         outstanding, then take the necessary
                                                                         steps to announce the subcontract)
              (COND
                ((NOT xpredecessors)
                  (ANNOUNCE!TASK xpnode (fetch (SUBCONTRACT NAME) of sc])
----------
Calls:     ANNOUNCE!TASK FIND!SUBCONTRACT SAME!STATUS!CHECK STORE!TASK!OBJECT

Freevars:  NET
```

Explanation: Called through CNET by a user task execution function. Generates a subtask of the contract with name "xname" in node "xpnode". The subtask is "xsubtask". The names of any subtasks that must be completed before this subtask can be announced are given by "predecessors".
A task object is created for the task, and a subcontract record is added to the subcontract list for the contract (the name is formed by consing the index of the subcontract with the name of the contract from which it is generated (i.e., a count (index) is kept of how many subcontracts have been generated from a contract). Subcontracts are not currently stored as separate objects.
Finally the subtask is announced.

rgs: 10-Oct-78 22:32 [CNET] GET!TASK!ANNOUNCEMENT

```
(GET!TASK!ANNOUNCEMENT
  [LAMBDA (xpnode xname)                                                  (* rgs: "10-Oct-78 22:32")
    (PROG (sc xsubtaskname xannproc)
          (SETQ sc (CAR (NODE!SEARCH xpnode xname 'ANNOUNCED NIL T)))
          (SETQ xsubtaskname (fetch (SUBCONTRACT TASK) of sc))            (* create a task announcement for the
                                                                          subtask)

          (* if a task!announcement!procedure is available, then use it to create the addressee,
          eligibility!specification, task!abstraction, bid!specification, and expiration!time -
          otherwise default to broadcast, NIL for the eligibility!specification, task!abstraction, and
          bid!specification, and default the expiration!time)


          (SETQ xannproc (fetch (TASK ANNOUNCEMENT!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK xsubtaskname)))
          (COND
            [xannproc (RETURN (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xannproc))
                                     (LIST xpnode (fetch (TASK SPECIFICATION) of (RETRIEVE!OBJECT xpnode 'TASK
                                                                                                    xsubtaskname]
            (T (RETURN (LIST "*" NIL (LIST (fetch (TASK TYPE) of (RETRIEVE!OBJECT xpnode 'TASK xsubtaskname)))
                             NIL tra])
```

Calls: NODE!SEARCH RETRIEVE!OBJECT

Called by: ANNOUNCE!TASK MAKE!BID

Freevars: tra

Explanation: Forms the essence of a task announcement for the (sub)contract with name "xname" in node "xpnode". If an 'announcement!procedure' exists for the task of the contract, then it is used to return a list of 'addressee', 'eligibility!specification', 'task!abstraction', 'bid!specification' and 'expiration!time'. Otherwise default values are used - "*" (for broadcast), NIL, the task 'type' (as a list), NIL, and tra (the default expiration time).

```
(GOOD!BOARD
  [LAMBDA (xboard)                                                    (* rgs: "16-Oct-78 21:54")
    (PROG (c r)
          (SETQ c (fetch (BOARD COLUMN) of xboard))
          (SETQ r (ELT (fetch (BOARD Q) of xboard)
                       c))
          (RETURN (NOT (OR (ELT (fetch (BOARD A) of xboard)
                                r)
                           (ELT (fetch (BOARD B) of xboard)
                                (IPLUS c r))
                           (ELT (fetch (BOARD C) of xboard)
                                (IDIFFERENCE (IPLUS qsize r)
                                             c])
----------
```

Called by: EXTEND!BOARD

Freevars: qsize

Explanation: Returns T if the board "xboard" is a plausible partial solution for the N Queens problem.

```
(INITCIL
  [LAMBDA NIL                                                    (* rgs: "27-Oct-78 10:26")
    (PROG NIL
          [SETQQ CILGRAMMAR (((#verb!phrase #noun!phrase #object!phrase)
                    (T (LIST #verb!phrase #noun!phrase #object!phrase)))
                   ((#noun!phrase)
                    (T #noun!phrase))
                   ((#verb!phrase #noun!phrase)
                    (T (LIST #verb!phrase #noun!phrase)))
                   ((#verb!phrase)
                    (T #verb!phrase]
          [SETQQ ELSPECGRAMMAR (((#OBJECT #attval)
                    ((for x in #attval always (ATTRIBUTEP #OBJECT (CAR x)))
                     (PROG (xobject)
                           (SETQ xobject (RETRIEVE!OBJECT xpnode #OBJECT (CADAR #attval)
                                                          (CAAR #attval)))
                           (RETURN (EQUAL (CADADR #attval)
                                          (RECORDACCESS (CAADR #attval)
                                                        xobject
                                                        (RECLOOK #OBJECT]
          [SETQQ TABSGRAMMAR (((#OBJECT (#attval))
                    ((for x in #attval always (ATTRIBUTEP #OBJECT (CAR x)))
                     (CONS #OBJECT #attval]
          [SETQQ BSPECGRAMMAR (((#VERB)
                    (T (LIST #VERB)))
                   ((#OBJECT #att1)
                    (T (COND
                         ((for x in #att1 always (ATTRIBUTEP #OBJECT x))
                           (CONS #OBJECT #att1]
          [SETQQ REPGRAMMAR (((#OBJECT (#attval))
                    ((for x in #attval ALWAYS (ATTRIBUTEP #OBJECT (CAR x)))
                     (CONS #OBJECT #attval)))
                   ((#VALUE)
                    (T (LIST #VALUE]
          (SETQ REQGRAMMAR NIL)
          [SETQQ INFOGRAMMAR (((#OBJECT (#attval))
                    ((for x in #attval ALWAYS (ATTRIBUTEP #OBJECT (CAR x)))
                     (CONS #OBJECT #attval]
          [PUTPROP '#ADJECTIVE 'POSSIBLEVALUES (COPY '(BUSY EVERY OWN]
          [PUTPROP '#ATTRIBUTE 'POSSIBLEVALUES
                    (COPY '(ANNOUNCEMENT!PROCEDURE ANNOUNCEMENT!RANKING!PROCEDURE AREA AWARD!PROCEDURE
                                                   BID!CONSTRUCTION!PROCEDURE BID!RANKING!PROCEDURE CODE DEVICE
                                                   EXECUTION!PROCEDURE INFORMATION!ACCEPTANCE!PROCEDURE LAT LONG MANAGER
                                                   NAME NUMBER POSITION PREDECESSOR REFUSAL!PROCEDURE
                                                   REFUSAL!PROCESSING!PROCEDURE RELATED!CONTRACTOR REPORT!RECIPIENT
                                                   RESULT SUBCONTRACT SUCCESSOR SPECIFICATION TYPE]
          (PUTPROP '#AUXILIARY 'POSSIBLEVALUES '(MUST))
          (PUTPROP '#CONNECTIVE 'POSSIBLEVALUES '(AND NOT OR))
          [PUTPROP '#OBJECT 'POSSIBLEVALUES (COPY '(CONTRACT DEVICE NODE POSITION PROCEDURE TASK TASK!TEMPLATE]
          (PUTPROP '#PREPOSITION 'POSSIBLEVALUES '(TO FROM WITH))
          (PUTPROP '#VALUE 'PREDICATE 'VALUEP)
          (PUTPROP '#VERB 'POSSIBLEVALUES '(ACKNOWLEDGE BID CHANGE CONFIRM HAVE REQUIRE RESPOND SEND SUSPEND))
          [PUTPROP '#att1 'GRAMMARS '(((#ATTRIBUTE #att1)
                      (T (CONS #ATTRIBUTE #att1)))
                     ((#ATTRIBUTE)
                      (T (LIST #ATTRIBUTE]
          [PUTPROP '#att2 'GRAMMARS '([(#att1 #obval #att2)
                      (T (CONS (APPEND #att1 #obval)
                               (LIST #att2]
                     ((#att1 #obval)
                      (T (APPEND #att1 #obval]
          [PUTPROP '#attval 'GRAMMARS '(((#ATTRIBUTE #VALUE #attval)
                      (T (CONS (LIST #ATTRIBUTE #VALUE)
                               #attval)))
                     ((#ATTRIBUTE #VALUE)
                      (T (LIST (LIST #ATTRIBUTE #VALUE]
          [PUTPROP '#noun!phrase 'GRAMMARS '(((#np2 #CONNECTIVE #np3)
                 (T (LIST #np2 #CONNECTIVE #np3)))
                ((#np1)
                 (T #np1))
                ((#ADJECTIVE #np1)
                 (T (LIST #ADJECTIVE #np1]
      [PUTPROP '#np1 'GRAMMARS '(((#OBJECT #att2)
                 (T (LIST #OBJECT #att2)))
                ((#OBJECT #att1)
                 ((AND (OBJECTP #OBJECT)
                       (for x in #att1 always (ATTRIBUTEP #OBJECT x)))
                  (LIST #OBJECT #att1)))
                ((#OBJECT)
                 ((OBJECTP #OBJECT)
                  #OBJECT))
                ((#VALUE (T #VALUE]
      [PUTPROP '#np2 'GRAMMARS '(((#np1)
                 (T #np1))
                ((#ADJECTIVE #np1)
                 (T (LIST #ADJECTIVE #np1]
      [PUTPROP '#np3 'GRAMMARS '(((#np1)
                 (T #np1))
                ((#ADJECTIVE #np1)
                 (T (LIST #ADJECTIVE #np1]
      [PUTPROP '#object!phrase 'GRAMMARS '(((#PREPOSITION #noun!phrase)
                 (T (LIST #PREPOSITION #noun!phrase]
      [PUTPROP '#obval 'GRAMMARS '(((#OBJECT)
                 (T (LIST #OBJECT)))
                ((#VALUE)
                 (T (LIST #VALUE]
      (PUTPROP '#verb!phrase 'GRAMMARS '(((#VERB)
                 (T #VERB))
                ((#AUXILIARY #VERB)
                 (T (LIST #AUXILIARY #VERB]))
-----------
```

Called by: INITIALIZE

Freevars: BSPECGRAMMAR CILGRAMMAR ELSPECGRAMMAR INFOGRAMMAR REPGRAMMAR REQGRAMMAR TABSGRAMMAR

Explanation: Initializes the common internode language.

```
(INITIALIZE
  [LAMBDA NIL                                                     (* rgs: "25-Oct-78 02:17")
    (PROG NIL
          (INITCIL)
          (SETQ eventlist (create EVENT TIME ← infinity DATA ←"header"))
          (SETQQ time -1)
          (SETQQ task!time 0)
          (SETQQ rtime 0)
          (SETQ NET (ARRAY netsize 0 NIL))
          (for xpnode from 1 to netsize do (SETA NET xpnode (create PNODE UTILIZATION ← 0 STATUS ←"Idle" KNOWLEDGE!BASE
                                                                     ←(create KNOWLEDGE!BASE)
                                                                     TASKCOUNTER ← 0))
                                           (STORE!OBJECT xpnode 'NODE (create NODE NAME ←'SELF)))
          (SETQ utilization (ARRAY netsize netsize))
          (SETQ eventcounter 0)
          (SETQ messagecounter 0)
          (SETQ bdcstcounter 0)
          (SETQ tacounter 0)
          (SETQ tracounter 0)
          (SETQ bidcounter 0)
          (SETQ aacounter 0)
          (SETQ dacounter 0)
          (SETQ acccounter 0)
          (SETQ recounter 0)
          (SETQ ircounter 0)
          (SETQ frcounter 0)
          (SETQ tecounter 0)
          (SETQ nacounter 0)
          (SETQ rqcounter 0)
          (SETQ imcounter 0])
----------
```

Calls: INITCIL STORE!OBJECT

Called by: SIMULATE

Freevars: NET aacounter acccounter bdcstcounter bidcounter dacounter eventcounter eventlist frcounter imcounter infinity ircounter messagecounter nacounter netsize recounter rqcounter rtime tacounter task!time tecounter time tracounter utilization

Explanation: Initializes the event list, time variables, nodes in the net, and node utilization.
The nodes are records in an array called NET. The index of the array corresponds to the name of the node.

rgs: 7-Sep-78 05:43 [CNET]   INSTALL!DISPLAY!EVENT

```
(INSTALL!DISPLAY!EVENT
  [LAMBDA (xtime xpnode xtype xdata)                              (* rgs: " 7-Sep-78 05:43")
    (PROG (d)
          (SETQ d (create DISPLAY!EVENT PNODE ← xpnode TYPE ← xtype DATA ← xdata))
          (INSTALL!EVENT xtime d])
----------
```

Calls: INSTALL!EVENT

Called by: NEXT!CONTRACT PROCESS!CONTRACT PROCESS!TERMINATION SDISPLAY TERMINATE!SUBCONTRACTS UPDATE!NODE

Explanation: Installs a display event in the event list at time "xtime". The event is placed by node "xpnode". It is of type "xtype" and the data of the event is given by "xdata".
There are two types of display event: 'simulation' and 'task'. Both are used to display text of some sort in the execution trace of the simulation. Simulation display events originate in contract net functions, and are used to display messages about the status of contract execution. Task display events originate in user functions and can be used to display any string at the correct simulation time, with an indicator as to the originator of the event.

```
rgs: 10-Sep-78 13:35 [CNET]                                                   INSTALL!EVENT
                                                                           -----------------

(INSTALL!EVENT
  [LAMBDA (xtime xdata)                                              (* rgs: "10-Sep-78 13:35")
    (PROG (e ev)
          (SETQ ev (create EVENT TIME ← xtime DATA ← xdata))
          (SETQ e eventlist)
          (while T do (COND
                         [(ILESSP xtime (fetch (EVENT TIME) of e))
                           (COND
                             ((fetch (EVENT LLINK) of e)
                               (SETQ e (fetch (EVENT LLINK) of e)))
                             (T (replace (EVENT LLINK) of e with ev)
                                (GO $$OUT]
                         (T (COND
                              ((fetch (EVENT RLINK) of e)
                                (SETQ e (fetch (EVENT RLINK) of e)))
                              (T (replace (EVENT RLINK) of e with ev)
                                 (GO $$OUT])
----------
Called by: INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT SENDMESSAGE

Freevars:  eventlist
```

Explanation: Installs an event in the event list at time "xtime". "xdata" is the data of the event.
The event list is currently stored as a binary tree. There are three types of event: messages, internal events, and display events.

```
rgs:  7-Sep-78 05:44 [CNET]                                          INSTALL!INTERNAL!EVENT
                                                                  --------------------------

(INSTALL!INTERNAL!EVENT
  [LAMBDA (xtime xpnode xname xtype xdata)                           (* rgs: " 7-Sep-78 05:44")
    (PROG (i)
          (SETQ i (create INTERNAL!EVENT PNODE ← xpnode NAME ← xname TYPE ← xtype DATA ← xdata))
          (INSTALL!EVENT xtime i])
----------
Calls:     INSTALL!EVENT

Called by: ANNOUNCE!TASK MAKE!BID NEXT!CONTRACT PROCESS!ANNOUNCED!AWARD PROCESS!CONTRACT PROCESS!DIRECTED!AWARD
           SIMULATE UPDATE!NODE
```

Explanation: Installs an internal event in the event list at time "xtime". The node involved is "xpnode", and the name of the contract involved is "xname". "xtype" is the 'type' of internal event and "xdata" is the data.
There are currently four types of internal event: 'contract!processing' and 'node!update', that are used to perform the necessary bookkeeping for task execution; 'bid!check', that is used to assess bids and take action (if necessary) at the end of the expiration time for a task announcement; and, 'pseudo!contract', that is used to eliminate (if necessary) the temporary information stored by a node in anticipation of the receipt of a contract on which a bid has been made.

```
(INTERIM!REPORT
  [LAMBDA (xpnode xname xrslt xaddressee)                                  (* edit: "18-Sep-78 07:37")
    (COND
      ((SAME!STATUS!CHECK xpnode xname)
        (PROG (xpnode! xcontract xreport)
              (SETQ xpnode! (ELT NET xpnode))
              (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT xname))
              (COND
                ((NOT xaddressee)
                  (SETQ xaddressee (fetch (CONTRACT REPORT!RECIPIENTS) of xcontract))
                                                                           (* default addressee is the list of
                                                                           report recipients for the contract)
                  ))
              (SETQ xreport (create INTERIM!REPORT NAME ← xname RESULT!DESCRIPTION ← xrslt))
              (SENDMESSAGE xpnode (IPLUS time tr2)
                           xaddressee xreport])
----------
Calls:     RETRIEVE!OBJECT SAME!STATUS!CHECK SENDMESSAGE

Freevars:  NET time tr2
```

Explanation: Sends an interim report from "xpnode" to "xaddressee" for the contract with name "xname". If "xaddressee" is NIL, then the report is sent to the report!recipients for the contract. The text of the report is "xrslt".

```
(MAKE!BID
  [LAMBDA (xpnode)                                                    (* rgs: "16-Oct-78 17:28")
    (PROG (xpnode! active pc oldest xan temp xtype $$es $$bs $$expt)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!))   (* currently bid on the oldest task when
                                                                              multiple task types are on the
                                                                              active!task!announcements list)
          (SETQ oldest (CAR active))
          (COND
            (oldest [for x in (CDR active) do (COND
                                                ((ILESSP (fetch (ACTIVE!TASK!ANNOUNCEMENT TIME) of x)
                                                         (fetch (ACTIVE!TASK!ANNOUNCEMENT TIME) of oldest))
                                                  (SETQ oldest x]
                    (BID xpnode (fetch (ACTIVE!TASK!ANNOUNCEMENT CONTRACT) of oldest)
                         (fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest)
                         (fetch (ACTIVE!TASK!ANNOUNCEMENT TYPE) of oldest)
                         (fetch (ACTIVE!TASK!ANNOUNCEMENT BID!SPECIFICATION) of oldest))
                    (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode! with (REMOVE oldest active))

          (* store a "pseudo-contract" in the knowledge base so that an award message can be handled without
          need for retransmission of task type and so on -
          then set up a "pseudo-contract" internal event to remove the pseudo-contract after the expiration
          time plus the time it takes to get an award has passed -
          that is, if the node is not awarded the contract)

                                                                              (* note that the "task" slot is filled
                                                                              in with a task!template "type" until the
                                                                              award is received)
                    (SETQ pc (create CONTRACT NAME ←(fetch (ACTIVE!TASK!ANNOUNCEMENT CONTRACT) of oldest)
                                     MANAGER ←(fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest)
                                     REPORT!RECIPIENTS ←(LIST (fetch (ACTIVE!TASK!ANNOUNCEMENT MANAGER) of oldest))
                                     TASK ←(fetch (ACTIVE!TASK!ANNOUNCEMENT TYPE) of oldest)
                                     STATE ←'PSEUDO))
                    (STORE!OBJECT xpnode 'CONTRACT pc)
                    (INSTALL!INTERNAL!EVENT (IPLUS time (fetch (ACTIVE!TASK!ANNOUNCEMENT EXPIRATION!TIME) of oldest)
                                                   tpb tsaw tpsaw)
                                            xpnode
                                            (fetch (CONTRACT NAME) of pc)
                                            'PSEUDO!CONTRACT))
            (T                                                                (* look on the announced list for
                                                                              outstanding subcontracts and make a bid
                                                                              on the oldest one for which the node
                                                                              meets the eligibility specification)
               (SETQ xan (REVERSE (fetch (PNODE ANNOUNCED) of xpnode!)))
               (COND
                 (xan [SETQ temp (CAAR (SOME xan (FUNCTION (LAMBDA (x)
                                                   (PROG (templ)
                                                         [SETQ templ (GET!TASK!ANNOUNCEMENT xpnode
                                                                                            (fetch (SUBCONTRACT NAME)
                                                                                               of (CAR x]
                                                         (COND
                                                           [(NOT (EQUAL (CAR templ)
                                                                        'DIRECTED!AWARD))
                                                             (SETQ $$es (CADR templ))
                                                             (COND
                                                               ((AND (NOT (fetch (SUBCONTRACT PREDECESSORS)
                                                                                 of (CAR x)))
                                                                     (OR (NOT $$es)
                                                                         (CHECK!ELIGIBILITY xpnode $$es)))
                                                                 (SETQ $$bs (CADDDR templ))
                                                                 (SETQ $$expt (CAR (CDDDDR templ)))
                                                                 (RETURN T))
                                                               (T (RETURN]
                                                           (T (RETURN]
                      (COND
                        (temp [SETQ xtype (fetch (TASK TYPE) of (RETRIEVE!OBJECT xpnode 'TASK
                                                                                   (fetch (SUBCONTRACT TASK) of temp]
                                  (BID xpnode (fetch (SUBCONTRACT NAME) of temp)
                                       xpnode xtype $$bs)
                                  (SETQ pc (create CONTRACT NAME ←(fetch (SUBCONTRACT NAME) of temp)
                                                   MANAGER ← xpnode REPORT!RECIPIENTS ←(LIST xpnode)
                                                   TASK ← xtype STATE ←'PSEUDO))
                                  (STORE!OBJECT xpnode 'CONTRACT pc)
                                  (INSTALL!INTERNAL!EVENT (IPLUS time $$expt tpb tsaw tpsaw)
                                                          xpnode
                                                          (fetch (SUBCONTRACT NAME) of temp)
                                                          'PSEUDO!CONTRACT]))
----------
```

Calls: BID CHECK!ELIGIBILITY GET!TASK!ANNOUNCEMENT INSTALL!INTERNAL!EVENT RETRIEVE!OBJECT STORE!OBJECT

Called by: NEXT!CONTRACT PROCESS!TASK!ANNOUNCEMENT

Freevars: NET time tpb tpsaw tsaw

Explanation: Makes a bid on an appropriate contract by node "xpnode".
First looks at the 'active!task!announcements' list. If only one task 'type' exists then bid on it. Otherwise bid on the oldest announcement. If no active task announcements exist then check the 'announced' state, and bid on the oldest subcontract for which the node meets the eligibility specification (and for which there are no predecessors).

rgs: 16-Oct-78 21:55 [CNET] NEW!BOARD

```
(NEW!BOARD
  [LAMBDA (xcol xq xa xb xc)                                          (* rgs: "16-Oct-78 21:55")
    (PROG (xcolumn xQ xA xB xC)
          (SETQ xcolumn (COND
              (xcol xcol)
              (T 1)))
          [SETQ xQ (COND
              (xq (COPYALL xq))
              (T (ARRAY qsize qsize]
          [SETQ xA (COND
              (xa (COPYALL xa))
              (T (ARRAY qsize NIL NIL]
          [SETQ xB (COND
              (xb (COPYALL xb))
              (T (ARRAY (LSH qsize 1)
                        NIL NIL]
          [SETQ xC (COND
              (xc (COPYALL xc))
              (T (ARRAY (SUB1 (LSH qsize 1))
                        NIL NIL]
          (RETURN (create BOARD COLUMN ← xcolumn Q ← xQ A ← xA B ← xB C ← xC])
----------
```

Called by: EXTEND!BOARD QINITIALIZE

Freevars: qsize

Explanation: Generates a new board for the N Queens problem. "xcol" is the column in which the next queen is to be placed (1 if "xcol" is NIL). "xq" is the array of current row indices in which queens have been placed (all NIL if "xq" is NIL). "xa", "xb", and "xc" are the arrays associated with Floyd's solution of the problem [JACM 14:4 Oct. '67, pp. 636-644] (all NIL if corresponding arguments are NIL).

```
(NEXT!CONTRACT
  [LAMBDA (xpnode xtype)                                            (* rgs: "23-Sep-78 16:44")
    (PROG (xpnode! rc)
          (SETQ xpnode! (ELT NET xpnode))
          (COND
            [(fetch (PNODE READY) of xpnode!)
              (SETQ rc (CAR (fetch (PNODE READY) of xpnode!)))
              (replace (PNODE EXECUTING) of xpnode! with (LIST (CAR rc)))
              (replace (PNODE READY) of xpnode! with (CDR (fetch (PNODE READY) of xpnode!]
            (T (replace (PNODE EXECUTING) of xpnode! with NIL)))
          (COND
            [(fetch (PNODE EXECUTING) of xpnode!)
              (replace (PNODE STATUS) of xpnode! with "Busy")
              (replace (CONTRACT STATE) of (CAR (fetch (PNODE EXECUTING) of xpnode!)) with 'EXECUTING)
              (COND
                [(CDR rc)
                  [INSTALL!DISPLAY!EVENT (IPLUS time (COND
                                                       ((EQUAL xtype 'REPORT)
                                                         tpr)
                                                       (T tt)))
                                         xpnode
                                         (fetch (CONTRACT NAME) of (CAR rc))
                                         'SIMULATION
                                         (APPEND '(Resumed Processing Contract)
                                                 (fetch (CONTRACT NAME) of (CAR rc]
                  (INSTALL!INTERNAL!EVENT (IPLUS time (COND
                                                        ((EQUAL xtype 'REPORT)
                                                          tpr)
                                                        (T tt)))
                                          xpnode
                                          (fetch (CONTRACT NAME) of (CAR rc))
                                          'NODE!UPDATE
                                          (LIST (CDR rc]
                (T [INSTALL!DISPLAY!EVENT (IPLUS time (COND
                                                        ((EQUAL xtype 'REPORT)
                                                          tpr)
                                                        (T tt)))
                                          xpnode
                                          (fetch (CONTRACT NAME) of (CAR rc))
                                          'SIMULATION
                                          (APPEND '(Started Processing Contract)
                                                  (fetch (CONTRACT NAME) of (CAR rc]
                   (INSTALL!INTERNAL!EVENT (IPLUS time (COND
                                                         ((EQUAL xtype 'REPORT)
                                                           tpr)
                                                         (T tt)))
                                           xpnode
                                           (fetch (CONTRACT NAME) of (CAR rc))
                                           'CONTRACT!PROCESSING]
            (T (replace (PNODE STATUS) of xpnode! with "Idle")
               (UPDATE!ACTIVE!TASK!ANNOUNCEMENTS xpnode)
               (MAKE!BID xpnode])
----------
```

Calls: INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT MAKE!BID UPDATE!ACTIVE!TASK!ANNOUNCEMENTS

Called by: PROCESS!FINAL!REPORT PROCESS!INTERIM!REPORT UPDATE!NODE

Freevars: NET time tpr tt

Explanation: Tries to install a new contract in the 'executing' state of node "xpnode". If a contract exists in the 'ready' state, then it is installed, and an appropriate event (either 'contract!processing' or 'node!update') is installed on the event list to start processing. Otherwise an attempt is made to bid on an active task announcement or outstanding subcontract. There are two types of call to this function, specified by "xtype": 'report' and 'termination'. The type is used to update simulation time in the correct manner.

```
(NEXT!EVENT
  [LAMBDA NIL
    (PROG (e1 e2)
          (SETQ e1 eventlist)
          (SETQ e2 (fetch (EVENT LLINK) of e1))
          [while (fetch (EVENT LLINK) of e2) do ((SETQ e1 e2)
                                                  (SETQ e2 (fetch (EVENT LLINK) of e2]
          (replace (EVENT LLINK) of e1 with (fetch (EVENT RLINK) of e2))
          (replace (EVENT LLINK) of e2 with NIL)
          (replace (EVENT RLINK) of e2 with NIL)
          (RETURN e2])
----------
```

Called by: SIMULATE

Freevars: eventlist

Explanation: Returns the next event to be processed from the event list.

```
(NODE!SEARCH
  [LAMBDA (xpnode xname xstate xdeleteflag xconditions)         (* rgs: "27-Sep-78 19:25")
                                                                (* for contracts in the executing and
                                                                terminated states the contract record is
                                                                returned)
                                                                (* for contracts in the ready and
                                                                suspended states the contract record is
                                                                returned -
                                                                bound to the possibilities list pointer)
                                                                (* for subcontracts in the announced
                                                                state the subcontract record is returned
                                                                -
                                                                bound to the active!bids)
    (PROG (xpnode! c)
          (SETQ xpnode! (ELT NET xpnode))
          (SELECTQ xstate
                   (EXECUTING (GO LEX))
                   (READY (GO LRD))
                   (ANNOUNCED (GO LAN))
                   (SUSPENDED (GO LSU))
                   (TERMINATED (GO LTR))
                   NIL)
      LEX [SETQ c (CAR (SOME (fetch (PNODE EXECUTING) of xpnode!)
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL xname (fetch (CONTRACT NAME) of x]
          (COND
            (c [COND
                 ((AND xdeleteflag (NOT (MEMBER 'EXECUTING xconditions)))
                   (replace (PNODE EXECUTING) of xpnode! with (REMOVE c (fetch (PNODE EXECUTING) of xpnode!]
               (RETURN c)))
          (COND
            (xstate (RETURN NIL)))
      LRD [SETQ c (CAR (SOME (fetch (PNODE READY) of xpnode!)
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL xname (fetch (CONTRACT NAME) of (CAR x]
          (COND
            (c [COND
                 ((AND xdeleteflag (NOT (MEMBER 'READY xconditions)))
                   (replace (PNODE READY) of xpnode! with (REMOVE c (fetch (PNODE READY) of xpnode!]
               (RETURN c)))
          (COND
            (xstate (RETURN NIL)))
      LAN [SETQ c (CAR (SOME (fetch (PNODE ANNOUNCED) of xpnode!)
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL xname (fetch (SUBCONTRACT NAME) of (CAR x]
          (COND
            (c [COND
                 ((AND xdeleteflag (NOT (MEMBER 'ANNOUNCED xconditions)))
                   (replace (PNODE ANNOUNCED) of xpnode! with (REMOVE c (fetch (PNODE ANNOUNCED) of xpnode!]
               (RETURN c)))
          (COND
            (xstate (RETURN NIL)))
      LSU [SETQ c (CAR (SOME (fetch (PNODE SUSPENDED) of xpnode!)
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL xname (fetch (CONTRACT NAME) of (CAR x]
          (COND
            (c [COND
                 ((AND xdeleteflag (NOT (MEMBER 'SUSPENDED xconditions)))
                   (replace (PNODE SUSPENDED) of xpnode! with (REMOVE c (fetch (PNODE SUSPENDED) of xpnode!]
               (RETURN c)))
          (COND
            (xstate (RETURN NIL)))
      LTR [SETQ c (CAR (SOME (fetch (PNODE TERMINATED) of xpnode!)
                             (FUNCTION (LAMBDA (x)
                                 (EQUAL xname (fetch (CONTRACT NAME) of x]
          (COND
            (c [COND
                ((AND xdeleteflag (NOT (MEMBER 'TERMINATED xconditions)))
                  (replace (PNODE TERMINATED) of xpnode! with (REMOVE c (fetch (PNODE TERMINATED) of xpnode!]
              (RETURN c])
----------
```

Called by: AWARD CHECK!BIDS DIRECTED!AWARD GET!TASK!ANNOUNCEMENT PROCESS!BID PROCESS!FINAL!REPORT PROCESS!INTERIM!REPORT TERMINATE!SUBCONTRACTS

Freevars: NET

Explanation: Searches the contract processing states of "xpnode" for the contract with name "xname", and returns the contract record, if found. "xstate" can specify the state to be searched. If "xdeleteflag" is T then the contract is removed from the processing state in which it is found. "xconditions" is a list of states from which the contract should not be deleted. It overrides "xdeleteflag".

rgs: 17-Oct-78 00:10 [CNET] OBJECTP

```
(OBJECTP
  [LAMBDA (xobject)                                          (* rgs: "17-Oct-78 00:10")
    (COND
      ((RECLOOK xobject)
        T)
      (T (WRITE "CIL error: " xobject " is not a valid object")
         NIL])
----------
```

Called by: ATTRIBUTEP

Explanation: Returns T if "xobject" is a valid object; else WRITEs an error message and returns NIL.

rgs: 18-Sep-78 00:45 [CNET] OUTSTANDING!SUBCONTRACTS

```
(OUTSTANDING!SUBCONTRACTS
  [LAMBDA (xcontract)                                        (* rgs: "18-Sep-78 00:45")
    (CAR (fetch (CONTRACT SUBCONTRACTS) of xcontract])
----------
```

Called by: QRECEIVE

Explanation: Returns the number of subcontracts of "xcontract" that have not yet been completed.

rgs: 10-Sep-78 17:17 [CNET] PARSE!NODE!ABSTRACTION

```
(PARSE!NODE!ABSTRACTION
  [LAMBDA (xabstraction)                                     (* rgs: "10-Sep-78 17:17")
    xabstraction])
----------
```

Called by: PROCESS!BID

Explanation:

```
rgs: 10-Sep-78 11:05 [CNET]                                                      PARSE!TASK!ABSTRACTION
                                                                               --------------------------
(PARSE!TASK!ABSTRACTION
  [LAMBDA (xabstraction)                                                  (* rgs: "10-Sep-78 11:05")
    xabstraction])
----------
Called by: PROCESS!DIRECTED!AWARD PROCESS!TASK!ANNOUNCEMENT

Explanation:



rgs:  7-Sep-78 08:49 [CNET]                                                      PROCESS!ACKNOWLEDGEMENT
                                                                               ---------------------------
(PROCESS!ACKNOWLEDGEMENT
  [LAMBDA (xpnode xmessage)                                               (* rgs: " 7-Sep-78 08:49")
    NIL])
----------
Called by: PROCESS!MESSAGE

Explanation:



rgs:  1-Oct-78 17:41 [CNET]                                                      PROCESS!ANNOUNCED!AWARD
                                                                               ---------------------------
(PROCESS!ANNOUNCED!AWARD
  [LAMBDA (xpnode xmessage)                                               (* rgs: " 1-Oct-78 17:41")
    (PROG (xpnode! xsa xcontract xtaskname temp)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xsa (fetch (MESSAGE TEXT) of xmessage))                   (* fetch the pseudo-contract record for
                                                                          the award)
          (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT (fetch (ANNOUNCED!AWARD NAME) of xsa)))
          (SETQ xtaskname (STORE!TASK!OBJECT xpnode (fetch (CONTRACT TASK) of xcontract)
                                             (fetch (ANNOUNCED!AWARD TASK!SPECIFICATION) of xsa)))
          (replace (CONTRACT TASK) of xcontract with xtaskname)
          (COND
            ((EQUAL (fetch (PNODE STATUS) of xpnode!)
                    "Busy")
              (replace (CONTRACT STATE) of xcontract with 'READY)
              (READY!CONTRACT xpnode xcontract))
            (T (replace (CONTRACT STATE) of xcontract with 'EXECUTING)
               (replace (PNODE EXECUTING) of xpnode! with (LIST xcontract))
               (replace (PNODE STATUS) of xpnode! with "Busy")
               (INSTALL!INTERNAL!EVENT (IPLUS time tpsaw)
                                       xpnode
                                       (fetch (CONTRACT NAME) of xcontract)
                                       'CONTRACT!PROCESSING])
----------
Calls:     INSTALL!INTERNAL!EVENT READY!CONTRACT RETRIEVE!OBJECT STORE!TASK!OBJECT

Called by: PROCESS!MESSAGE

Freevars:  NET time tpsaw

Explanation:  Performs the necessary bokeeping to handle the receipt of an announced award by node "xpnode".
           "xmessage" is the message. If the node is "Idle", then an event is placed on the event list to begin
           processing on the new contract. Otherwise the contract is placed in the 'ready' state.
```

```
(PROCESS!BID
  [LAMBDA (xpnode xmessage)                                                    (* rgs: "27-Sep-78 19:29")
    (PROG (xpnode! xbid xnode!abstraction xbidl xbidrankproc sc active!bids)
          (SETQ xpnode! (ELT NET xpnode))

          (* make sure that the contract has not yet been awarded by searching for it in the announced state -
          remember that it is bound to the active!bids)


          (SETQ xbid (fetch (MESSAGE TEXT) of xmessage))
          (SETQ sc (NODE!SEARCH xpnode (fetch (BID NAME) of xbid)
                             'ANNOUNCED NIL T))
          (COND
            (sc (SETQ xnode!abstraction (PARSE!NODE!ABSTRACTION (fetch (BID NODE!ABSTRACTION) of xbid)))
                (SETQ xbidl (create ACTIVE!BID CONTRACTOR ←(fetch (MESSAGE ORIGINATOR) of xmessage)
                                    ABSTRACTION ← xnode!abstraction TIME ←(fetch (MESSAGE TIME) of xmessage)))
                [SETQ xbidrankproc (fetch (TASK BID!RANKING!PROCEDURE) of (RETRIEVE!OBJECT xpnode 'TASK
                                                                                   (fetch (SUBCONTRACT TASK)
                                                                                     of (CAR sc]
                                                                           (* if there is a bid ranking procedure
                                                                           then use it, else cons the new bid to
                                                                           the old active!bids list)
                [COND
                  (xbidrankproc [SETQ active!bids (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                       'PROCEDURE
                                                                                       xbidrankproc))
                                                        (LIST xbidl (CADR sc]
                                                                           (* the bid ranking procedure returns
                                                                           (action list). if the action is
                                                                           (QUOTE satisfactory,) then award the
                                                                           contract to the new bidder)
                                )
                  (T (SETQ active!bids (CONS xbidl (CADR sc]
                (COND
                  [(EQUAL (CAR active!bids)
                          'SATISFACTORY)
                    (AWARD xpnode (fetch (SUBCONTRACT NAME) of (CAR sc))
                           (fetch (ACTIVE!BID CONTRACTOR) of (CADR active!bids]
                  (T                                                       (* update the active!bids list)
                     (FRPLACA (CDR sc)
                              active!bids])
```

Calls: AWARD NODE!SEARCH PARSE!NODE!ABSTRACTION RETRIEVE!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET

Explanation: Performs the necessary bokeeping to handle the receipt of a bid by node "xpnode". "xmessage" is the message. If the contract has not already been awarded then it is ranked relative to other bids using the bid!ranking!procedure for the task. If no procedure exists, then the bid is consed to the old active!bid list for the contract.

If a bid!ranking!procedure exists, and it returns 'satisfactory as the first element of its list of values, then the contract is awarded to the contractor named in the active!bid record that is the second element of the list.

```
(PROCESS!CONTRACT
  [LAMBDA (xpnode xname)                                              (* rgs: " 6-Oct-78 20:40")

          (* to get the processing on the task associated with a contract started, set up a possibilities list
          and use TRYNEXT to get a value -
          the value is only used for messages to the update!node function which determines whether a contract
          should be suspended, terminated, or resumed)


    (PROG (xpnode! xcontract taskprocesspointer temp)
          (SETQ xpnode! (ELT NET xpnode))
          (COND
            ((EQUAL (fetch (PNODE STATUS) of xpnode!)
                    "Idle")
              (RETURN))
            (T (SETQ xcontract (CAR (fetch (PNODE EXECUTING) of xpnode!)))
               (INSTALL!DISPLAY!EVENT time xpnode 'SIMULATION (APPEND '(Started Processing Contract)
                                                                      xname))
               [SETQ taskprocesspointer (POSSIBILITIES (APPLY [fetch (PROCEDURE CODE)
                                                               of (RETRIEVE!OBJECT
                                                                    xpnode
                                                                    'PROCEDURE
                                                                    (fetch (TASK EXECUTION!PROCEDURE)
                                                                       of (RETRIEVE!OBJECT xpnode 'TASK
                                                                                  (fetch (CONTRACT TASK)
                                                                                     of xcontract]
                                                            (LIST xpnode (fetch (CONTRACT NAME) of xcontract)
                                                                  (fetch (TASK SPECIFICATION)
                                                                     of (RETRIEVE!OBJECT xpnode 'TASK
                                                                                 (fetch (CONTRACT TASK)
                                                                                    of xcontract)))
                                                                  xcontract]
               (SETQ temp (RESUME!TASK taskprocesspointer))
               (INSTALL!INTERNAL!EVENT (IPLUS time task!time)
                                       xpnode
                                       (fetch (CONTRACT NAME) of xcontract)
                                       'NODE!UPDATE
                                       (CONS taskprocesspointer temp))
               (RETURN T])
----------
```

Calls: INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT RESUME!TASK RETRIEVE!OBJECT

Called by: PROCESS!INTERNAL!EVENT

Freevars: NET task!time time

Explanation: Starts the processing of the contract named "xname" in node "xpnode". Sets up the task execution function as a generator via POSSIBILITIES. Then installs a 'node!update' event to continue, after performing a RESUME!TASK.

```
(PROCESS!DIRECTED!AWARD
  [LAMBDA (xpnode xmessage)                                            (* rgs: " 1-Oct-78 18:07")
    (PROG (xabs xpnode! xcontract xda xtaskname xtt xrefproc xrefjust)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xda (fetch (MESSAGE TEXT) of xmessage))
          (SETQ xabs (PARSE!TASK!ABSTRACTION (fetch (DIRECTED!AWARD TASK!ABSTRACTION) of xda)))
          (SETQ xtt (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (CAR xabs)))
          (COND
            ((AND (CHECK!ELIGIBILITY xpnode (fetch (DIRECTED!AWARD ELIGIBILITY!SPECIFICATION) of xda))
                  xtt)
              (SETQ xtaskname (STORE!TASK!OBJECT xpnode (CAR xabs)
                                                 (fetch (DIRECTED!AWARD TASK!SPECIFICATION) of xda)))
              (SETQ xcontract (create CONTRACT NAME ←(fetch (DIRECTED!AWARD NAME) of xda)
                                      MANAGER ←(fetch (MESSAGE ORIGINATOR) of xmessage)
                                      REPORT!RECIPIENTS ←(LIST (fetch (MESSAGE ORIGINATOR) of xmessage))
                                      TASK ← xtaskname))
              (STORE!OBJECT xpnode 'CONTRACT xcontract)
              [COND
                ((EQUAL (fetch (PNODE STATUS) of xpnode!)
                        "Busy")
                  (replace (CONTRACT STATE) of xcontract with 'READY)
                  (READY!CONTRACT xpnode xcontract))
                (T (replace (CONTRACT STATE) of xcontract with 'EXECUTING)
                   (replace (PNODE EXECUTING) of xpnode! with (LIST xcontract))
                   (replace (PNODE STATUS) of xpnode! with "Busy")
                   (INSTALL!INTERNAL!EVENT (IPLUS time tpdaw)
                                           xpnode
                                           (fetch (CONTRACT NAME) of xcontract)
                                           'CONTRACT!PROCESSING]
              (SENDMESSAGE xpnode (IPLUS time tpdaw tack)
                           (fetch (CONTRACT MANAGER) of xcontract)
                           (create ACKNOWLEDGEMENT NAME ←(fetch (CONTRACT NAME) of xcontract)
                                   RESPONSE ←'ACCEPTANCE)))
            (T [SETQ xrefproc (COND
                   (xtt (fetch (TASK!TEMPLATE REFUSAL!PROCEDURE) of xtt]
               (SETQ xrefjust (COND
                   (xrefproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xrefproc))
                                    (LIST xpnode xda)))
                   (T NIL)))
               (SENDMESSAGE xpnode (IPLUS time tpdaw tnack)
                            (fetch (MESSAGE ORIGINATOR) of xmessage)
                            (create ACKNOWLEDGEMENT NAME ←(fetch (DIRECTED!AWARD NAME) of xda)
                                    RESPONSE ←'REFUSAL
                                    REFUSAL!JUSTIFICATION ← xrefjust])
----------
```

Calls: CHECK!ELIGIBILITY INSTALL!INTERNAL!EVENT PARSE!TASK!ABSTRACTION READY!CONTRACT RETRIEVE!OBJECT SENDMESSAGE STORE!OBJECT STORE!TASK!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET tack time tnack tpdaw

Explanation: Performs the necessary bookeeping to handle the receipt of a directed award by node "xpnode". "xmessage" is the message. If the node meets the eligibility specification for the task and has a task template for task type mentioned in the task abstraction, then the contract is accepted and affirmatively acknowledged. Otherwise the contract is refused and a negative acknowledgement is sent to the originator. the 'refusal!justification' is obtained from the procedure for the task, or set to NIL, if no procedure exists.

If the node is "Idle", then an event is placed on the event list to begin processing on the new contract. Otherwise the contract is placed in the 'ready' state.

```
(PROCESS!DISPLAY!EVENT
  [LAMBDA (d)                                                   (* rgs: " 7-Sep-78 05:47")
    (DISPLAY "From: " (fetch (DISPLAY!EVENT PNODE) of d))
    (DISPLAY)
    (DISPLAY (fetch (DISPLAY!EVENT DATA) of d))
    (DISPLAY])
----------
```

Called by: DISPLAY!EVENT SIMULATE

Explanation: Displays the data of display event "d" with an indication about its originator.

```
(PROCESS!FINAL!REPORT
  [LAMBDA (xpnode xmessage)                                        (* rgs: "27-Sep-78 21:54")
    (PROG (xpnode! pname xcontract xsubcontract xstate xrepaccproc sc nsc xreport tmpsc tmpscl)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xreport (fetch (MESSAGE TEXT) of xmessage))
          (SETQ pname (CDR (fetch (FINAL!REPORT NAME) of xreport)))
          (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT pname))
          (COND
            (xcontract (SETQ xstate (fetch (CONTRACT STATE) of xcontract))
                       (COND
                         ((NOT (EQUAL xstate 'TERMINATED))
                           [SETQ nsc (SUB1 (CAR (fetch (CONTRACT SUBCONTRACTS) of xcontract]
                           [SETQ tmpsc (CAR (SOME (CDR (fetch (CONTRACT SUBCONTRACTS) of xcontract))
                                                  (FUNCTION (LAMBDA (x)
                                                              (EQUAL (fetch (SUBCONTRACT NAME) of x)
                                                                     (fetch (FINAL!REPORT NAME) of xreport]
                           [for x in (fetch (SUBCONTRACT SUCCESSORS) of tmpsc)
                              do (SETQ tmpscl (FIND!SUBCONTRACT xpnode x))
                                 (replace (SUBCONTRACT PREDECESSORS) of tmpscl
                                    with (REMOVE (fetch (SUBCONTRACT NAME) of tmpsc)
                                                 (fetch (SUBCONTRACT PREDECESSORS) of tmpscl]
                           [for x in (fetch (SUBCONTRACT PREDECESSORS) of tmpsc)
                              do (SETQ tmpscl (FIND!SUBCONTRACT xpnode x))
                                 (replace (SUBCONTRACT SUCCESSORS) of tmpscl
                                    with (REMOVE (fetch (SUBCONTRACT NAME) of tmpsc)
                                                 (fetch (SUBCONTRACT SUCCESSORS) of tmpscl]
                           (RPLACA (fetch (CONTRACT SUBCONTRACTS) of xcontract)
                                   nsc)
                           [RPLACD (fetch (CONTRACT SUBCONTRACTS) of xcontract)
                                   (REMOVE tmpsc (CDR (fetch (CONTRACT SUBCONTRACTS) of xcontract]
                           (COND
                             ((EQUAL nsc 0)
                               (replace (CONTRACT SUBCONTRACTS) of xcontract with NIL)))
                           (SETQ xrepaccproc (fetch (TASK REPORT!ACCEPTANCE!PROCEDURE) of (RETRIEVE!OBJECT
                                                                                            xpnode
                                                                                            'TASK xcontract)))
                           [COND
                             [xrepaccproc (replace (CONTRACT RESULTS) of xcontract
                                             with (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                                      'PROCEDURE
                                                                                                      xrepaccproc))
                                                         (LIST (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport)
                                                               (fetch (CONTRACT RESULTS) of xcontract]
                             (T (replace (CONTRACT RESULTS) of xcontract
                                   with (COND
                                          ((fetch (CONTRACT RESULTS) of xcontract)
                                            (APPEND (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport)
                                                    (fetch (CONTRACT RESULTS) of xcontract)))
                                          (T (fetch (FINAL!REPORT RESULT!DESCRIPTION) of xreport]
                           (COND
                             ((EQUAL xstate 'SUSPENDED)
                               (SETQ sc (NODE!SEARCH xpnode (fetch (CONTRACT NAME) of xcontract)
                                                     'SUSPENDED T))
                               (replace (PNODE READY) of xpnode! with (SORT (CONS sc (fetch (PNODE READY) of xpnode!))
                                                                             'READYCOMPARE))
                               (replace (CONTRACT STATE) of (CAR sc) with 'READY)
                               (COND
                                 ((EQUAL (fetch (PNODE STATUS) of xpnode!)
                                         "Idle")
                                   (NEXT!CONTRACT xpnode 'REPORT])
----------
Calls:     FIND!SUBCONTRACT NEXT!CONTRACT NODE!SEARCH READYCOMPARE RETRIEVE!OBJECT

Called by: PROCESS!MESSAGE

Freevars:  NET
```

Explanation: Performs the necessary bookkeeping to handle the receipt of a final report by node "xpnode". "xmessage" is the message. If the contract for which the report is intended has not been terminated, then the appropriate subcontract is deleted from the list of subcontracts for the contract. Predecessors and successors are updated. The 'report!acceptance!procedure' for the contract is used to update the 'results' slot. If no procedure exists then the new result is appended to the previous results.
If the contract is currently in the 'suspended' state, then it is moved to the 'ready' state.
The status of the node is checked, and another contract executed if the node is "Idle".

rgs: 11-Oct-78 08:06 [CNET] PROCESS!INFORMATION

```
(PROCESS!INFORMATION
  [LAMBDA (xpnode xmessage)                                           (* rgs: "11-Oct-78 08:06")
    (PROG (xpnode! pname xcontract xstate xinfoaccproc xinfo)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xinfo (fetch (MESSAGE TEXT) of xmessage))
          (SETQ pname (fetch (INFORMATION NAME) of xinfo))
          (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT pname))
          (COND
            (xcontract (SETQ xstate (fetch (CONTRACT STATE) of xcontract))
                       (COND
                         ((NOT (EQUAL xstate 'TERMINATED))
                           (SETQ xinfoaccproc (fetch (TASK INFORMATION!ACCEPTANCE!PROCEDURE)
                                                     of (RETRIEVE!OBJECT xpnode 'TASK xcontract)))
                           (COND
                             [xinfoaccproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE
                                                                                                xinfoaccproc))
                                                  (LIST (fetch (INFORMATION INFORMATION!SPECIFICATION) of xinfo]
                             (T (for x in (fetch (INFORMATION INFORMATION!SPECIFICATION) of xinfo)
                                   do (STORE!OBJECT xpnode (CAR x)
                                                    (EVAL (CONS 'create
                                                                (PROG (z)
                                                                      (SETQ z (CAR x))
                                                                      [for y in (CDR x)
                                                                         do (SETQ z (APPEND z
                                                                                            (CONS (CAR y)
                                                                                                  (CONS '←(CDR y]
                                                                      (RETURN z])
```

Calls: RETRIEVE!OBJECT STORE!OBJECT

Called by: PROCESS!MESSAGE

Freevars: NET

Explanation:

```
(PROCESS!INTERIM!REPORT
  [LAMBDA (xpnode xmessage)                                        (* rgs: "23-Sep-78 16:45")
    (PROG (xpnode! pname xcontract xsubcontract xstate xrepaccproc sc xreport)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xreport (fetch (MESSAGE TEXT) of xmessage))
          (SETQ pname (CDR (fetch (INTERIM!REPORT NAME) of xreport)))
          (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT pname))
          (COND
            (xcontract (SETQ xstate (fetch (CONTRACT STATE) of xcontract))
                       (COND
                         ((NOT (EQUAL xstate 'TERMINATED))
                           (replace (SUBCONTRACT RESULTS) of [CAR (SOME (CDR (fetch (CONTRACT SUBCONTRACTS)
                                                                                      of xcontract))
                                                                        (FUNCTION (LAMBDA (x)
                                                                            (EQUAL (fetch (SUBCONTRACT NAME) of x)
                                                                                   (fetch (INTERIM!REPORT NAME)
                                                                                     of xreport]
                             with (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport))
                           (SETQ xrepaccproc (fetch (TASK REPORT!ACCEPTANCE!PROCEDURE) of (RETRIEVE!OBJECT
                                                                                              xpnode
                                                                                              'TASK xcontract)))
                           [COND
                             [xrepaccproc (replace (CONTRACT RESULTS) of xcontract
                                                  with (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                                          'PROCEDURE
                                                                                                          xrepaccproc))
                                                              (LIST (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)
                                                                    (fetch (CONTRACT RESULTS) of xcontract]
                             (T (replace (CONTRACT RESULTS) of xcontract
                                      with (COND
                                             ((fetch (CONTRACT RESULTS) of xcontract)
                                               (APPEND (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport)
                                                       (fetch (CONTRACT RESULTS) of xcontract)))
                                             (T (fetch (INTERIM!REPORT RESULT!DESCRIPTION) of xreport]
                           (COND
                             ((EQUAL xstate 'SUSPENDED)
                               (SETQ sc (NODE!SEARCH xpnode (fetch (CONTRACT NAME) of xcontract)
                                                     'SUSPENDED T))
                               (replace (PNODE READY) of xpnode! with (SORT (CONS sc (fetch (PNODE READY) of xpnode!))
                                                                            'READYCOMPARE))
                               (replace (CONTRACT STATE) of (CAR sc) with 'READY)
                               (COND
                                 ((EQUAL (fetch (PNODE STATUS) of xpnode!)
                                         "Idle")
                                   (NEXT!CONTRACT xpnode 'REPORT])
----------
Calls:     NEXT!CONTRACT NODE!SEARCH READYCOMPARE RETRIEVE!OBJECT

Called by: PROCESS!MESSAGE

Freevars:  NEXT
```

Explanation: Performs the necessary bookeeping to handle the receipt of an interim report by node "xpnode". "xmessage" is the message. If the contract for which the report is intended has not been terminated, then the 'results' slot of the appropriate subcontract is updated with the new result.

The 'report!acceptance!procedure' for the contract is used to update the 'results' slot. If no procedure exists then the new result is appended to the previous results.

If the contract is currently in the 'suspended' state, then it is moved to the 'ready' state.

The status of the node is checked, and another contract executed if the node is "Idle".

rgs: 6-Oct-78 20:39 [CNET] **PROCESS!INTERNAL!EVENT**

```
(PROCESS!INTERNAL!EVENT
  [LAMBDA (e)                                                                (* rgs: " 6-Oct-78 20:39")
    (PROG NIL
          (RETURN (SELECTQ (fetch (INTERNAL!EVENT TYPE) of e)
                           (CONTRACT!PROCESSING (PROCESS!CONTRACT (fetch (INTERNAL!EVENT PNODE) of e)
                                                                  (fetch (INTERNAL!EVENT NAME) of e)))
                           (NODE!UPDATE (UPDATE!NODE (fetch (INTERNAL!EVENT PNODE) of e)
                                                     (fetch (INTERNAL!EVENT NAME) of e)
                                                     (fetch (INTERNAL!EVENT DATA) of e)))
                           (BID!CHECK (CHECK!BIDS (fetch (INTERNAL!EVENT PNODE) of e)
                                                  (fetch (INTERNAL!EVENT NAME) of e)))
                           (PSEUDO!CONTRACT (DELETE!PSEUDO!CONTRACT (fetch (INTERNAL!EVENT PNODE) of e)
                                                                    (fetch (INTERNAL!EVENT NAME) of e)))
                           NIL])
----------
Calls:     CHECK!BIDS DELETE!PSEUDO!CONTRACT PROCESS!CONTRACT UPDATE!NODE

Called by: SIMULATE
```

Explanation: Routes internal event "e" to the appropriate function.
There are currently four types of internal event: 'contract!processing' and 'node!update', that are used to perform the necessary bookeeping for task execution; 'bid!check', that is used to assess bids and take action (if necessary) at the end of the expiration time for a task announcement; and, 'pseudo!contract', that is used to eliminate (if necessary) the temporary information stored by a node in anticipation of the receipt of a contract on which a bid has been made.

rgs: 17-Oct-78 21:23 [CNET] **PROCESS!MESSAGE**

```
(PROCESS!MESSAGE
  [LAMBDA (xpnode xaddressee xmessage)                                       (* rgs: "17-Oct-78 21:23")
    (PROG NIL
          (COND
            ((OR (EQUAL xaddressee "*")
                 (EQUAL xpnode xaddressee)
                 (MEMBER xpnode xaddressee))
              (SELECTQ (CAR (fetch (MESSAGE TEXT) of xmessage))
                       (TASK!ANNOUNCEMENT (PROCESS!TASK!ANNOUNCEMENT xpnode xmessage))
                       (BID (PROCESS!BID xpnode xmessage))
                       (ANNOUNCED!AWARD (PROCESS!ANNOUNCED!AWARD xpnode xmessage))
                       (DIRECTED!AWARD (PROCESS!DIRECTED!AWARD XPNODE XMESSAGE))
                       (ACKNOWLEDGEMENT (PROCESS!ACKNOWLEDGEMENT xpnode xmessage))
                       (INTERIM!REPORT (PROCESS!INTERIM!REPORT xpnode xmessage))
                       (FINAL!REPORT (PROCESS!FINAL!REPORT xpnode xmessage))
                       (TERMINATION (PROCESS!TERMINATION xpnode xmessage))
                       (REQUEST (PROCESS!REQUEST xpnode xmessage))
                       (INFORMATION (PROCESS!INFORMATION xpnode xmessage))
                       (NODE!AVAILABILITY!ANNOUNCEMENT (PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT xpnode xmessage))
                       NIL])
----------
Calls:     PROCESS!ACKNOWLEDGEMENT PROCESS!ANNOUNCED!AWARD PROCESS!BID PROCESS!DIRECTED!AWARD PROCESS!FINAL!REPORT
           PROCESS!INFORMATION PROCESS!INTERIM!REPORT PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT PROCESS!REQUEST
           PROCESS!TASK!ANNOUNCEMENT PROCESS!TERMINATION

Called by: SIMULATE

Freevars:  XMESSAGE XPNODE
```

Explanation: Routes the message "xmessage" to node "xpnode" if it is one of the addressees (or the message is a broadcast). The addressee is "xaddressee".
The message is routed to the appropriate function. There is a function to receive each of the messages of the contract net protocol.

rgs: 7-Sep-78 05:53 [CNET]   PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT

```
(PROCESS!NODE!AVAILABILITY!ANNOUNCEMENT
  [LAMBDA (xpnode xmessage)                                        (* rgs: " 7-Sep-78 05:53")
    NIL])
----------
```

Called by: PROCESS!MESSAGE

Explanation:

rgs: 10-Oct-78 23:27 [CNET]   PROCESS!REQUEST

```
(PROCESS!REQUEST
  [LAMBDA (xpnode xmessage)                                        (* rgs: "10-Oct-78 23:27")
    (PROG (xpnode! pname xrequest xreqspec xobject xinfo)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xrequest (fetch (MESSAGE TEXT) of xmessage))
          (SETQ pname (fetch (REQUEST NAME) of xrequest))
          (SETQ xreqspec (fetch (REQUEST REQUEST!SPECIFICATION) of xrequest))
          [SETQ xinfo (for x in xreqspec collect xobject when (SETQ xobject (RETRIEVE!OBJECT xpnode (CAR x)
                                                                                          (CADDR x)
                                                                                          (CADR x]
          (COND
            (xinfo (SENDMESSAGE xpnode time (fetch (MESSAGE ORIGINATOR) of xmessage)
                                (create INFORMATION NAME ← pname INFORMATION!SPECIFICATION ← xinfo])
----------
```

Calls: RETRIEVE!OBJECT SENDMESSAGE

Called by: PROCESS!MESSAGE

Freevars: NET time

Explanation:

```
(PROCESS!TASK!ANNOUNCEMENT
  [LAMBDA (xpnode xmessage)                                             (* rgs: "10-Oct-78 22:40")
    (PROG (xpnode! active xta xabs xtt sametype xtal xarankproc pc rank)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!))
          (SETQ xta (fetch (MESSAGE TEXT) of xmessage))
          (UPDATE!ACTIVE!TASK!ANNOUNCEMENTS xpnode)
          (SETQ xabs (PARSE!TASK!ABSTRACTION (fetch (TASK!ANNOUNCEMENT TASK!ABSTRACTION) of xta)))
          (SETQ xtt (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (CAR xabs)))
          [COND
            ([AND xtt (OR (NOT (fetch (TASK!ANNOUNCEMENT ELIGIBILITY!SPECIFICATION) of xta))
                          (CHECK!ELIGIBILITY xpnode (fetch (TASK!ANNOUNCEMENT ELIGIBILITY!SPECIFICATION) of xta]
              [SETQ sametype (CAR (SOME active (FUNCTION (LAMBDA (x)
                                      (EQUAL (fetch (ACTIVE!TASK!ANNOUNCEMENT TYPE) of x)
                                             (CAR xabs]
              (COND
                [sametype

          (* if there is an active task announcement of the same type as the new task announcement then get
          the announcement!ranking!procedure and apply it to determine if the old active task announcement
          should be replaced -
          if there is no ranking procedure then keep the old active task announcement)


                          (SETQ xarankproc (fetch (TASK!TEMPLATE ANNOUNCEMENT!RANKING!PROCEDURE) of xtt))
                          (COND
                            (xarankproc

          (* the announcement!ranking!procedure is passed the parsed abstraction for the new task announcement
          and the parsed task abstraction for the old active task announcment of the same type -
          it returns 1 if the new announcement is "better", 0 if the two are equally important, and -1 if the
          old active task announcement is "better" -
          the current default if they are equally important is to retain the old active task announcement)


                                          [SETQ rank (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                                    'PROCEDURE
                                                                                                    xarankproc))
                                                            (LIST (CADR xabs)
                                                                  (fetch (ACTIVE!TASK!ANNOUNCEMENT ABSTRACTION)
                                                                    of sametype]
                                          (COND
                                            ((EQUAL rank 1)
                                              (SETQ xtal (create ACTIVE!TASK!ANNOUNCEMENT MANAGER ←(fetch (MESSAGE
                                                                                                          ORIGINATOR)
                                                                                                    of xmessage)
                                                                 CONTRACT ←(fetch (TASK!ANNOUNCEMENT NAME) of xta)
                                                                 TYPE ←(CAR xabs)
                                                                 ABSTRACTION ←(CADR xabs)
                                                                 BID!SPECIFICATION ←(fetch (TASK!ANNOUNCEMENT
                                                                                              BID!SPECIFICATION)
                                                                                     of xta)
                                                                 TIME ←(fetch (MESSAGE TIME) of xmessage)
                                                                 EXPIRATION!TIME ←(fetch (TASK!ANNOUNCEMENT
                                                                                            EXPIRATION!TIME)
                                                                                   of xta)))
                                              (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!
                                                 with (SUBST xtal sametype active]
                (T (SETQ xtal (create ACTIVE!TASK!ANNOUNCEMENT MANAGER ←(fetch (MESSAGE ORIGINATOR) of xmessage)
                                      CONTRACT ←(fetch (TASK!ANNOUNCEMENT NAME) of xta)
                                      TYPE ←(CAR xabs)
                                      ABSTRACTION ←(CADR xabs)
                                      BID!SPECIFICATION ←(fetch (TASK!ANNOUNCEMENT BID!SPECIFICATION) of xta)
                                      TIME ←(fetch (MESSAGE TIME) of xmessage)
                                      EXPIRATION!TIME ←(fetch (TASK!ANNOUNCEMENT EXPIRATION!TIME) of xta)))
                   (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!
                      with (CONS xtal (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!]
          (COND
            ((EQUAL (fetch (PNODE STATUS) of xpnode])
                    "Idle")
              (MAKE!BID xpnode]))
----------
```

Calls: CHECK!ELIGIBILITY MAKE!BID PARSE!TASK!ABSTRACTION RETRIEVE!OBJECT UPDATE!ACTIVE!TASK!ANNOUNCEMENTS

Called by: PROCESS!MESSAGE

Freevars: NET

Explanation: Performs the necessary bookeeping to handle the receipt of a task announcement by node "xpnode". "xmessage" is the message. If the node meets the eligibility specification of the task then the task announcement is ranked relative to other currently active task announcements. If there are other announcements of the same type (as specified by the task!abstraction), then the 'announcement!ranking!procedure' is used to rank them. If there is no procedure, then the old announcement is kept, and the new one is discarded. If there are no other announcements of the same type then the new announcement is consed to the list of other announcements.

The announcement!ranking!procedure returns +1, 0, or -1. +1 indicates that the new announcement should be kept, -1 that the old announcement should be kept, and 0 if the two are equally good (the current default is to keep the old one in this case).

If the node is "Idle", then a bid is made on the current best task announcement.

```
(PROCESS!TERMINATION
  [LAMBDA (xpnode xmessage)                                          (* rgs: "16-Oct-78 09:13")
    (PROG (xpnode! xterm xcontract xtn xta xtt xtermproc)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xterm (fetch (MESSAGE TEXT) of xmessage))
          (SETQ xcontract (RETRIEVE!OBJECT xpnode 'CONTRACT (fetch (TERMINATION NAME) of xterm)))
          (COND
            ((NOT (EQUAL (fetch (CONTRACT STATE) of xcontract)
                         'TERMINATED))
              (replace (PNODE TERMINATED) of xpnode! with (CONS xcontract (fetch (PNODE TERMINATED) of xpnode!)))
              (replace (CONTRACT STATE) of xcontract with 'TERMINATED)
              [COND
                ((ILESSP ntermcs (LENGTH (fetch (PNODE TERMINATED) of xpnode!)))
                  (DREVERSE (fetch (PNODE TERMINATED) of xpnode!))
                  [SETQ xtn (fetch (CONTRACT TASK) of (CAR (fetch (PNODE TERMINATED) of xpnode!]
                  (SETQ xta (RETRIEVE!OBJECT xpnode 'TASK xtn))
                  [SETQ xtermproc (fetch (TASK TERMINATION!PROCEDURE)
                                    of (RETRIEVE!OBJECT xpnode 'TASK (fetch (CONTRACT TASK)
                                                                      of (CAR (fetch (PNODE TERMINATED) of xpnode!]
                  [COND
                    [xtermproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode 'PROCEDURE xtermproc))
                                      (LIST xpnode (CAR (fetch (PNODE TERMINATED) of xpnode!]
                    (T (SETQ xtt (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (fetch (TASK TYPE) of xta)))
                       (replace (TASK!TEMPLATE TASKS) of xtt with (REMOVE xtn (fetch (TASK!TEMPLATE TASKS) of xtt)))
                       (DELETE!OBJECT xpnode 'TASK xtn)
                       (DELETE!OBJECT xpnode 'CONTRACT (fetch (CONTRACT NAME) of (CAR (fetch (PNODE TERMINATED)
                                                                                    of xpnode!]
                  (replace (PNODE TERMINATED) of xpnode! with (DREVERSE (CDR (fetch (PNODE TERMINATED) of xpnode!]
              (INSTALL!DISPLAY!EVENT (IPLUS time tpt)
                                     xpnode
                                     'SIMULATION
                                     (APPEND '(Terminated Contract)
                                             (fetch (TERMINATION NAME) of xterm)))
                                                                     (* now terminate the subcontracts)
              (TERMINATE!SUBCONTRACTS xpnode xcontract tpt])
----------
Calls:      DELETE!OBJECT INSTALL!DISPLAY!EVENT RETRIEVE!OBJECT TERMINATE!SUBCONTRACTS

Called by: PROCESS!MESSAGE

Freevars:  NET ntermcs time tpt
```

Explanation: Performs the necessary bookeeping to handle the receipt of a termination by node "xpnode". "xmessage" is the message. If the contract named in the message has not already been terminated then it is placed in the terminated state, and all of its outstanding subcontracts are terminated.
If the terminated state contains more than 'ntermcs' contracts then the oldest contract is discarded, after presenting it to the 'termination!procedure' for its task (if such a procedure exists).

```
(QANNOUNCE
  [LAMBDA (xpnode xspecification)                                    (* rgs: "16-Oct-78 21:55")
    (LIST "*" NIL (LIST 'EXTEND!BOARD (SUB1 (fetch (BOARD COLUMN) of xspecification)))
          NIL tra])
----------
Freevars:  tra
```

Explanation: Returns the essence of a task announcement for the N Queens 'extend board' task. "xpnode" is the node and "xspecification" is the task specification. The abstraction is a list of the task type and the number of queens already placed.

```
rgs: 15-Sep-78 10:10 [CNET]                                                              QARANK
                                                                                       ----------
(QARANK
  [LAMBDA (xabs1 xabs2)                                                   (* rgs: "15-Sep-78 10:10")
                                                                          (* rank the two abstractions on the
                                                                          basis of the selected "local" search
                                                                          strategy)
    (PROG (rn)
          (RETURN (SELECTQ qsearch!strategy
                           (0 (COND
                                ((IGREATERP xabs1 xabs2)
                                  1)
                                ((IGREATERP xabs2 xabs1)
                                  -1)
                                (T 0)))
                           (1 (COND
                                ((IGREATERP xabs2 xabs1)
                                  1)
                                ((IGREATERP xabs1 xabs2)
                                  -1)
                                (T 0)))
                           (2 (SETQ rn (RAND -1.0 1.0))
                              (COND
                                ((FGREATERP rn 0.0)
                                  1)
                                ((MINUSP rn)
                                  -1)
                                (T 0)))
                           NIL])
----------
Freevars:  qsearch!strategy

Explanation:  Orders two 'extend board' task abstractions, "xabs1" (the abstraction for the 'new' announcement) and
              "xabs2" (the abstration for the current best announcement). Returns +1, -1, or 0 according to the search
              strategy determined in QSET!PARAMETERS.


rgs: 12-Sep-78 00:11 [CNET]                                                              QBRANK
                                                                                       ----------
(QBRANK
  [LAMBDA (newbid oldbids)                                                (* rgs: "12-Sep-78 00:11")
    (LIST 'SATISFACTORY newbid])
----------
Explanation:  Handles bids received by a node. "newbid" is the 'new' active!bid, and "oldbids" is the list of
              previously received active!bids. QBRANK always returns a list of 'satisfactory and the new active!bid.


rgs: 16-Sep-78 03:51 [CNET]                                                              QDISPLAY
                                                                                       ------------
(QDISPLAY
  [LAMBDA (xboard)                                                        (* rgs: "16-Sep-78 03:51")
    (CONS "Queen-rows:" (for i from 1 to qsize collect (ELT xboard i) when (IGREATERP (ELT xboard i)
                                                                                       0])
----------
Called by: EXTEND!BOARD QFINALIZE

Freevars:  qsize

Explanation:  Returns a list of rows in which queens have been placed on the board "xboard".
```

```
edit: 18-Sep-78 07:01 [CNET]                                                                         QFINALIZE
                                                                                                     -------------
(QFINALIZE
  [LAMBDA (xpnode xname xrslt)                                          (* edit: "18-Sep-78 07:01")
    (PROG (solutions msg)
          [SETQ solutions (for x in xrslt collect x when (EQUAL (CAR x)
                                                                 'SUCCESS]
          (COND
            (solutions (SETQ msg (LIST "Solutions Found:
"))
                       [for x in solutions do (SETQ msg (APPEND msg (QDISPLAY (CADDDR x))
                                                                 (LIST "
"]
                       (DISPLAY msg))
            (T (DISPLAY "No Solutions Found"])
----------
Calls:     QDISPLAY
```

Explanation: The 'final!function' for the N Queens problem. Displays the solutions found (or that no solutions have been found).
"xpnode" is the node that sent the top-level report. "xname" is the name of the contract. "xrslt" is the text of the report.

```
rgs: 16-Oct-78 21:56 [CNET]                                                                        QINITIALIZE
                                                                                                   ---------------
(QINITIALIZE
  [LAMBDA (xnetsize restartflag olduserparamflag)                       (* rgs: "16-Oct-78 21:56")
    (PROG (xprocedure xannproc xarankproc xbrankproc xtask!template)
          [COND
            ((NOT restartflag)
              (QSET!PARAMETERS (NOT olduserparamflag]
          (SETQ xprocedure (create PROCEDURE NAME ←'EXTEND!BOARD
                                   CODE ←'EXTEND!BOARD))
          (SETQ xannproc (create PROCEDURE NAME ←'QANNOUNCE
                                 CODE ←'QANNOUNCE))
          (SETQ xarankproc (create PROCEDURE NAME ←'QARANK
                                   CODE ←'QARANK))
          (SETQ xbrankproc (create PROCEDURE NAME ←'QBRANK
                                   CODE ←'QBRANK))
          (SETQ xtask!template (create TASK!TEMPLATE TYPE ←'EXTEND!BOARD
                                       ANNOUNCEMENT!PROCEDURE ←'QANNOUNCE
                                       ANNOUNCEMENT!RANKING!PROCEDURE ←'QARANK
                                       BID!RANKING!PROCEDURE ←'QBRANK
                                       EXECUTION!PROCEDURE ←'EXTEND!BOARD))
          (for x from 1 to xnetsize do (STORE!OBJECT x 'PROCEDURE xprocedure)
                                       (STORE!OBJECT x 'PROCEDURE xannproc)
                                       (STORE!OBJECT x 'PROCEDURE xarankproc)
                                       (STORE!OBJECT x 'PROCEDURE xbrankproc)
                                       (STORE!OBJECT x 'TASK!TEMPLATE (COPYALL xtask!template)))
          (RETURN (LIST (LIST 'EXTEND!BOARD (NEW!BOARD])
----------
Calls:     NEW!BOARD QSET!PARAMETERS STORE!OBJECT
```

Explanation: The 'initial!function' for the N Queens problem. Initializes the knowledge bases of the nodes in the net with the required task!templates and procedures. Returns a list of the top-level task type and the initial board (no queens placed).
"xnetsize" is the number of nodes in the net. "restartflag" is T if new parameters are not to be requested. "olduserparamflag" is T if the current user parameters are to be used as defaults when new user parameters are requested.

rgs: 18-Sep-78 09:22 [CNET]   QRECEIVE

```
(QRECEIVE
  [LAMBDA (xpnode xname xcontract)                                    (* rgs: "18-Sep-78 09:22")
    (PROG (solutions)
          (COND
            ((EQUAL qreport!strategy 0)
              (while (AND (OUTSTANDING!SUBCONTRACTS xcontract)
                          (ILESSP [LENGTH (SETQ solutions (for x in (fetch (CONTRACT RESULTS) of xcontract)
                                                              collect x when (EQUAL (CAR x)
                                                                                    'SUCCESS]
                                  qnsol))
                 do [CNET* 'INTERIM!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract]
                    (SUSPEND))
              [CNET* 'FINAL!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract]
              (TERMINATE))
            (T (while (OUTSTANDING!SUBCONTRACTS xcontract) do (SUSPEND))
               [SETQ solutions (for x in (fetch (CONTRACT RESULTS) of xcontract) collect x
                                  when (EQUAL (CAR x)
                                              'SUCCESS]
               [COND
                 (solutions (CNET* 'FINAL!REPORT (LIST solutions)))
                 (T (CNET* 'FINAL!REPORT (LIST (LIST (CAR (fetch (CONTRACT RESULTS) of xcontract]
               (TERMINATE])
```

Calls: CNET* OUTSTANDING!SUBCONTRACTS SUSPEND TERMINATE

Called by: EXTEND!BOARD

Freevars: qnsol qreport!strategy

Explanation: Actually decides what to do upon receipt of a report for the N Queens problem. "xpnode" is the node receiving the report. "xname" is the name of the contract. "xcontract" is the contract record.

rgs: 23-Sep-78 18:32 [CNET]   QSET!PARAMETERS

```
(QSET!PARAMETERS
  [LAMBDA (cleanstart)                                                (* rgs: "23-Sep-78 18:32")
    (PROG NIL
          [COND
            (cleanstart (PROG NIL
                              (SETQQ qsize 5)
                              (SETQQ qnsol 1)
                              (SETQQ qsearch!strategy 0)
                              (SETQQ qreport!strategy 0)
                              (SETQQ tqgenerate 1)
                              (SETQQ tqsubtask 1)
                              (SETQQ tqsuccess 1)
                              (SETQQ tqfailure 1]
          (TTYOUT)
          (SETQ qsize (ASKFORNUMBER "Number of Queens" qsize 'QSIZE 0))
          (SETQ qsearch!strategy (ASKFORNUMBER "Search Strategy" qsearch!strategy 'QSEARCH -1 3))
          (SETQ qreport!strategy (ASKFORNUMBER "Report Strategy" qreport!strategy 'QREPORT -1 2))
          [COND
            ((EQUAL qreport!strategy 0)
              (SETQ qnsol (ASKFORNUMBER "Number of solutions" qnsol 'QNSOL 0]
          (TTYOUT])
```

Called by: QINITIALIZE

Freevars: qnsol qreport!strategy qsearch!strategy qsize tqfailure tqgenerate tqsubtask tqsuccess

Explanation: Asks user for parameters for the N Queens problem. Sets global variables. Same style as SET!PARAMETERS.
    If "cleanstart" is T then the settings built into the function are used as defaults for the questions.

rgs: 23-Sep-78 16:38 [CNET]   RANDOMCOMPARE

```
(RANDOMCOMPARE
  [LAMBDA (a b)                                                              (* rgs: "23-Sep-78 16:38")
    (COND
      ((FGREATERP (RAND -1.0 1.0)
                  0.0)
        T)
      (T NIL])
```

Called by: SIMULATE

Explanation: Orders two items "a", and "b" according to a random number between -1 and +1.

rgs: 18-Sep-78 01:57 [CNET]   READY!CONTRACT

```
(READY!CONTRACT
  [LAMBDA (xpnode xcontract xpointer)                                        (* rgs: "18-Sep-78 01:57")
    (PROG (xpnode!)
          (SETQ xpnode! (ELT NET xpnode))
          (RETURN (replace (PNODE READY) of xpnode! with (COND
                                                           [(fetch (PNODE READY) of xpnode!)
                                                             (NCONC (fetch (PNODE READY) of xpnode!)
                                                                    (LIST (CONS xcontract xpointer]
                                                           (T (LIST (CONS xcontract xpointer])
```

Called by: PROCESS!ANNOUNCED!AWARD PROCESS!DIRECTED!AWARD

Freevars: NET

Explanation: The contract with name "xname" is placed at the end of the list of contracts in the 'ready' state of node "xpnode". "xpointer" can be a pointer to the task execution procedure, if READY!CONTRACT is called to ready a suspended contract.

rgs: 19-Sep-78 17:28 [CNET]   READYCOMPARE

```
(READYCOMPARE
  [LAMBDA (a b)                                                              (* rgs: "19-Sep-78 17:28")
    (COND
      ((AND (CDR a)
            (CDR b))
        (COND
          ([ILESSP (LENGTH (fetch (CONTRACT NAME) of (CAR a)))
                   (LENGTH (fetch (CONTRACT NAME) of (CAR b]
            T)
          (T NIL)))
      ((AND (CDR a)
            (NOT (CDR b)))
        T)
      ((AND (NOT (CDR a))
            (CDR b))
        NIL)
      (T T])
```

Called by: PROCESS!FINAL!REPORT PROCESS!INTERIM!REPORT

Explanation: Orders two contracts in the ready state, "a", and "b". The ordering is such that resumed contracts have priority over newly acquired contracts, and the older resumed contracts have priority over the newer ones.

rgs: 27-Oct-78 10:51 [CNET]   REANNOUNCE!TASK

```
(REANNOUNCE!TASK
  [LAMBDA (xpnode xname)                                        (* rgs: "27-Oct-78 10:51")
    (PROG NIL
          (SETQ tracounter (ADD1 tracounter))
          (ANNOUNCE!TASK xpnode xname])
----------
```

Calls: ANNOUNCE!TASK

Called by: CHECK!BIDS

Freevars: tracounter

rgs: 10-Aug-78 15:58 [CNET]   RELEASE!TASK

```
(RELEASE!TASK
  [LAMBDA (taskprocesspointer)                                  (* rgs: "10-Aug-78 15:58")
    (TRYNEXT taskprocesspointer NIL 'RELEASE])
----------
```

Called by: UPDATE!NODE

Explanation: Used to release the pointer to a task execution function. It does this by calling up the function with the pointer "taskprocesspointer" with TRYNEXT, and passing the keyword 'RELEASE. The function that always catches this keyword is CNET*, and it performs an 'ADIEU.
   This function is used to release the pointer to a task when the associated contract is terminated by the manager.

rgs: 9-Jul-78 16:00 [CNET]   RESIMULATE

```
(RESIMULATE
  [LAMBDA NIL                                                   (* rgs: " 9-Jul-78 16:00")
    (PROG NIL
          (TTYOUT)
          (SETQ resimulateflag (ASKFORYESNO "Another task" resimulateflag 'RESTART))
          (COND
            ((NOT resimulateflag)
              (COND
                (fileflag (CLOSEF cnetfile)))
              (RETURN 0))
            (T (SETQ sameparameterflag (ASKFORYESNO "Same parameters" sameparameterflag 'RESTARTPARAMS))
               (COND
                 ((NOT sameparameterflag)
                   (COND
                     (fileflag (CLOSEF cnetfile)))
                   (RETURN 2))
                 (T (RETURN 1])
----------
```

Called by: CNET

Freevars: cnetfile fileflag resimulateflag sameparameterflag

Explanation: Asks the user questions about doing another simulation.
              Returns 0 if the user doesn't want another simulation to be done.
              Returns 1 if another simulation is to be done with the same parameters.
              Returns 2 if another simulation is to be done with different parameters.

```
rgs: 10-Aug-78 13:58 [CNET]                                                                 RESUME!TASK
                                                                                           ---------------
(RESUME!TASK
  [LAMBDA (taskprocesspointer)                                           (* rgs: "10-Aug-78 13:58")
    (SETQ task!time 0)
    (TRYNEXT taskprocesspointer])
----------
Called by: PROCESS!CONTRACT UPDATE!NODE

Freevars:  task!time

Explanation:  Resumes the task with pointer "taskprocesspointer" with TRYNEXT.  Also initializes the 'task!time'.
```

```
rgs: 20-Oct-78 15:22 [CNET]                                                             RETRIEVE!OBJECT
                                                                                       -------------------
(RETRIEVE!OBJECT
  [LAMBDA (xpnode xobject xkey xslot everyflag)                          (* rgs: "20-Oct-78 15:22")
    (PROG (xpnode! kb index otherindex)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ kb (fetch (PNODE KNOWLEDGE!BASE) of xpnode!))
          (COND
            [(MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE!BASE))
              [SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE]
              (RETURN (COND
                        [xkey (CAR (SOME index (FUNCTION (LAMBDA (x)
                                                   (EQUAL [COND
                                                            ((NOT xslot)
                                                              (CADR x))
                                                            (T (RECORDACCESS xslot x (RECLOOK xobject]
                                                          xkey]
                        (T (COND
                             (everyflag index)
                             (T (CAR index]
            (T (SETQ index (fetch (KNOWLEDGE!BASE OTHER) of kb))
               [SETQ otherindex (CDAR (SOME index (FUNCTION (LAMBDA (x)
                                                      (EQUAL (CAR x)
                                                             xobject]
               (RETURN (COND
                         [xkey (CAR (SOME otherindex (FUNCTION (LAMBDA (x)
                                                         (EQUAL [COND
                                                                  ((NOT xslot)
                                                                    (CADR x))
                                                                  (T (RECORDACCESS xslot x (RECLOOK xobject]
                                                                xkey]
                         (T (COND
                              (everyflag otherindex)
                              (T (CAR otherindex])
----------
Called by: AWARD BID CHECK!BIDS DELETE!PSEUDO!CONTRACT FINAL!REPORT FIND!SUBCONTRACT GET!TASK!ANNOUNCEMENT
           INTERIM!REPORT MAKE!BID PROCESS!ANNOUNCED!AWARD PROCESS!BID PROCESS!CONTRACT PROCESS!DIRECTED!AWARD
           PROCESS!FINAL!REPORT PROCESS!INFORMATION PROCESS!INTERIM!REPORT PROCESS!REQUEST PROCESS!TASK!ANNOUNCEMENT
           PROCESS!TERMINATION STORE!TASK!OBJECT UPDATE!NODE

Freevars:  NET

Explanation:  Returns the record for the object of type "xobject" in node "xpnode".  "xkey" is the key used to find
           the object. "xslot" is the name of the slot to which the key belongs. If "xslot" is NIL then the first slot
           for the object is used.
```

```
rgs:  7-Sep-78 06:02 [CNET]                                                             SAME!STATUS!CHECK
                                                                                        ------------------

(SAME!STATUS!CHECK
  [LAMBDA (xpnode xname)                                                 (* rgs: " 7-Sep-78 06:02")

          (* used to check that the node is still executing the same contract that it was when the event to be
          processed was placed on the event list)


    (AND (EQUAL (fetch (PNODE STATUS) of (ELT NET xpnode))
                "Busy")
         (EQUAL [fetch (CONTRACT NAME) of (CAR (fetch (PNODE EXECUTING) of (ELT NET xpnode]
                xname])
----------
Called by: FINAL!REPORT GENERATE!SUBTASK INTERIM!REPORT SDISPLAY UPDATE!NODE

Freevars:  NET

Explanation:  Returns T if "xpnode" iis "Busy", and the contract with name "xname" is being executed.



rgs:  7-Sep-78 06:02 [CNET]                                                                     SDISPLAY
                                                                                                ---------

(SDISPLAY
  [LAMBDA (xpnode xname xdata)                                           (* rgs: " 7-Sep-78 06:02")
    (COND
      ((SAME!STATUS!CHECK xpnode xname)
        (INSTALL!DISPLAY!EVENT time xpnode 'TASK xdata])
----------
Calls:     INSTALL!DISPLAY!EVENT SAME!STATUS!CHECK

Freevars:  time

Explanation:  Used to display text from a user task execution function in the trace of the simulation. "xpnode" is
           the originating node, "xname" is the name of the contract, and "xdata" is the text.
              This function is called through CNET* (which inserts the "xpnode" and "xname" arguments).



rgs: 12-Sep-78 01:02 [CNET]                                                                  SENDMESSAGE
                                                                                            ------------

(SENDMESSAGE
  [LAMBDA (xoriginator xtime xaddressee xtext)                           (* rgs: "12-Sep-78 01:02")
    (PROG (m)
          (SETQ m (create MESSAGE TIME ← xtime ADDRESSEE ← xaddressee ORIGINATOR ← xoriginator TEXT ← xtext))
          (INSTALL!EVENT xtime m])
----------
Calls:     INSTALL!EVENT

Called by: ANNOUNCE!TASK AWARD BID DIRECTED!AWARD FINAL!REPORT INTERIM!REPORT PROCESS!DIRECTED!AWARD PROCESS!REQUEST
           TERMINATE!SUBCONTRACTS

Explanation:  Sends a message from "xoriginator" to "xaddressee" at time "xtime".  "xtext" is the text of the
           message. The message is sent by placiing a message event on the event list.
```

```
(SET!PARAMETERS
  [LAMBDA (cleanstart)                                          (* rgs: "23-Sep-78 18:30")
    (PROG NIL
          [COND
            (cleanstart (PROG NIL
                              (SETQQ netsize 10)
                              (SETQQ gain 100)
                              (SETQQ ntermcs 10)
                              (SETQQ dpflag T)
                              (SETQQ dpfflag T)
                              (SETQQ delayfile cnet.delay)
                              (SETQQ ta 1)
                              (SETQQ tra 1000)
                              (SETQQ tpa 1)
                              (SETQQ tna 1)
                              (SETQQ tpna 1)
                              (SETQQ tb 1)
                              (SETQQ tpb 1)
                              (SETQQ tsaw 1)
                              (SETQQ tpsaw 1)
                              (SETQQ tdaw 1)
                              (SETQQ tpdaw 1)
                              (SETQQ tack 1)
                              (SETQQ tpack 1)
                              (SETQQ tr2 1)
                              (SETQQ tpr 1)
                              (SETQQ tt 1)
                              (SETQQ tpt 1)
                              (SETQQ treq 1)
                              (SETQQ tpreq 1)
                              (SETQQ ti 1)
                              (SETQQ tpi 1)
                              (SETQQ display!parameter!flag T)
                              (SETQQ display!statistics!flag T)
                              (SETQQ display!banners!flag T)
                              (SETQQ display!time!flag T)
                              (SETQQ display!messages!flag NIL)
                              (SETQQ display!internal!events!flag NIL)
                              (SETQQ display!display!events!flag NIL)
                              (SETQQ display!node!flag NIL)
                              (SETQQ fileflag NIL)
                              (SETQQ cnetfile cnet.results)
                              (SETQQ termflag T)
                              (SETQQ resimulateflag T)
                              (SETQQ sameparameterflag T)
                              (SETQQ initial!function $INITIALIZE)
                              (SETQQ final!function $FINALIZE)
                              (SETQ randstart (RANDSET T]
          (TTYOUT)
          (SETQ netsize (ASKFORNUMBER "Nodes" netsize 'NETSIZE 0))
          (SETQ nodelist (for i from 1 to netsize collect i))
          (SETQ gain (ASKFORNUMBER "Task time expansion factor" gain 'GAIN 0))
          (SETQ ntermcs (ASKFORNUMBER "Terminated contracts" ntermcs 'NTERMCS -1))
          (SETQ dpflag (ASKFORYESNO "Default delay parameters" dpflag 'DELAY))
          [COND
            ((NOT dpflag)
              (SETQ dpfflag (ASKFORYESNO "Read parameters from a file" dpfflag 'DELAYFILE))
              (COND
                (dpfflag (SETQ delayfile (ASKFORFILENAME 'INPUT delayfile))
                         (INPUT (INFILE delayfile))
                         (SETQ ta (READ delayfile))
                         (SETQ tra (READ delayfile))
                         (SETQ tpa (READ delayfile))
                         (SETQ tna (READ delayfile))
                         (SETQ tpna (READ delayfile))
                         (SETQ tb (READ delayfile))
                         (SETQ tpb (READ delayfile))
```

```
                    (SETQ tsaw (READ delayfile))
                    (SETQ tpsaw (READ delayfile))
                    (SETQ tdaw (READ delayfile))
                    (SETQ tpdaw (READ delayfile))
                    (SETQ tack (READ delayfile))
                    (SETQ tpack (READ delayfile))
                    (SETQ tr2 (READ delayfile))
                    (SETQ tpr (READ delayfile))
                    (SETQ tt (READ delayfile))
                    (SETQ tpt (READ delayfile))
                    (SETQ treq (READ delayfile))
                    (SETQ tpreq (READ delayfile))
                    (SETQ ti (READ delayfile))
                    (SETQ tpi (READ delayfile))
                    (CLOSEF delayfile))
              (T (SETQ ta (ASKFORNUMBER "ta" ta 'TA 0))
                 (SETQ tra (ASKFORNUMBER "tra" tra 'TRA 0))
                 (SETQ tpa (ASKFORNUMBER "tpa" tpa 'TPA 0))
                 (SETQ tna (ASKFORNUMBER "tna" tna 'TNA 0))
                 (SETQ tpna (ASKFORNUMBER "tpna" tpna 'TPNA 0))
                 (SETQ tb (ASKFORNUMBER "tb" tb 'TB 0))
                 (SETQ tpb (ASKFORNUMBER "tpb" tpb 'TPB 0))
                 (SETQ tsaw (ASKFORNUMBER "tsaw" tsaw 'TSAW 0))
                 (SETQ tpsaw (ASKFORNUMBER "tpsaw" tpsaw 'TPSAW 0))
                 (SETQ tdaw (ASKFORNUMBER "tdaw" tdaw 'TDAW 0))
                 (SETQ tpdaw (ASKFORNUMBER "tpdaw" tpdaw 'TPDAW 0))
                 (SETQ tack (ASKFORNUMBER "tack" tack 'TACK 0))
                 (SETQ tpack (ASKFORNUMBER "tpack" tpack 'TPACK 0))
                 (SETQ tr2 (ASKFORNUMBER "tr2" tr2 'TR2 0))
                 (SETQ tpr (ASKFORNUMBER "tpr" tpr 'TPR 0))
                 (SETQ tt (ASKFORNUMBER "tt" tt 'TT 0))
                 (SETQ tpt (ASKFORNUMBER "tpt" tpt 'TPT 0))
                 (SETQ treq (ASKFORNUMBER "treq" treq 'TREQ 0))
                 (SETQ tpreq (ASKFORNUMBER "tpreq" tpreq 'TPREQ 0))
                 (SETQ ti (ASKFORNUMBER "ti" ti 'TI 0))
                 (SETQ tpi (ASKFORNUMBER "tpi" tpi 'TPI 0]
          (SETQ display!parameter!flag (ASKFORYESNO "Display Parameters" display!parameter!flag 'DPARAM))
          (SETQ display!statistics!flag (ASKFORYESNO "Display statistics" display!statistics!flag 'DSTAT))
          (SETQ display!banners!flag (ASKFORYESNO "Display banners" display!banners!flag 'DBAN))
          (SETQ display!time!flag (ASKFORYESNO "Display time" display!time!flag 'DTIME))
          (SETQ display!messages!flag (ASKFORYESNO "Display messages" display!messages!flag 'DMESS))
          (SETQ display!internal!events!flag (ASKFORYESNO "Display internal events" display!internal!events!flag
                                                           'DINTE))
          (SETQ display!display!events!flag (ASKFORYESNO "Display display events" display!display!events!flag
                                                          'DDIS))
          (SETQ display!node!flag (ASKFORYESNO "Display nodes" display!node!flag 'DNODE))
          (SETQ display!events!flag (OR display!messages!flag display!internal!events!flag display!display!events!flag))
          (SETQ fileflag (ASKFORYESNO "Diagnostic information to file" fileflag 'OFILE))
          (SETQ initial!function (ASKFORFUNCTIONNAME "Initial Applications Function" initial!function
                                                      'INITIALIZE))
          (SETQ final!function (ASKFORFUNCTIONNAME "Final Applications Function" final!function 'FINALIZE))
          (COND
            (fileflag (SETQ cnetfile (ASKFORFILENAME 'OUTPUT cnetfile))
                      (SETQ termflag (ASKFORYESNO "Also to the terminal" termflag 'TERM))
                      (OUTFILE cnetfile)))
          (RANDSET randstart)
          (TTYOUT]))
----------
```

Called by: CNET

Freevars: cnetfile delayfile display!banners!flag display!display!events!flag display!events!flag display!internal!events!flag display!messages!flag display!node!flag display!parameter!flag display!statistics!flag display!time!flag dpfflag dpflag fileflag final!function gain initial!function netsize nodelist ntermcs randstart resimulateflag sameparameterflag ta tack tb tdaw termflag ti tna tpa tpack tpb tpdaw tpi tpna tpr tpreq tpsaw tpt tr2 tra treq tsaw tt

```
Explanation:  Asks the user for parameter settings for the simulation. Sets global variables.
              All questions give a prompt, have a default, and respond to "?" with a help message.
              If "cleanstart" is T then the settings built into the function are used as defaults for the questions.
```

```
(SIMULATE
  [LAMBDA (restartflag olduserparamflag)                                    (* rgs: "27-Oct-78 10:48")
    (PROG (xpnode xpnode! eventflag ev evdata xaddressee delta newtimeflag initial!tasks xcontract)
          (COND
            (display!parameter!flag (DISPLAY)
                                    (DISPLAY!PARAMETERS)
                                    (DISPLAY)))
          (COND
            (display!banners!flag (DISPLAY)
                                  (DISPLAY "::::::::::::::::::::::::::: Start of Simulation :::::::::::::::::::::::::::::::")
                                  (DISPLAY)))
          (INITIALIZE)
          (SETQ initial!tasks (APPLY initial!function (LIST netsize restartflag olduserparamflag)))
          (SETQ xpnode 0)
          (do (SETQ xpnode (ADD1 xpnode))
              (SETQ xpnode! (ELT NET xpnode))                               (* initial!function must initialize the
                                                                            knowledge bases of the nodes in the net
                                                                            and return a list of top level task
                                                                            names)
              [SETQ xcontract (create CONTRACT NAME ←(LIST xpnode)
                                      MANAGER ← 0 REPORT!RECIPIENTS ←(LIST 0)
                                      TASK ←(STORE!TASK!OBJECT xpnode (CAAR initial!tasks)
                                                               (CADAR initial!tasks]
              (STORE!OBJECT xpnode 'CONTRACT xcontract)
              (replace (PNODE EXECUTING) of xpnode! with (LIST xcontract))
              (replace (PNODE STATUS) of xpnode! with "Busy")
              (INSTALL!INTERNAL!EVENT 0 xpnode xpnode 'CONTRACT!PROCESSING)
              (SETQ initial!tasks (CDR initial!tasks)) until (NULL initial!tasks))
                                                                            (* should also add new contracts to
                                                                            knowledge base (also new tasks))
          (do (SETQ eventflag NIL)
              (SETQ ev (NEXT!EVENT))
              (SETQ eventcounter (ADD1 eventcounter))
              (COND
                ((IGREATERP (fetch (EVENT TIME) of ev)
                            time)
                  (SETQ delta (IDIFFERENCE (fetch (EVENT TIME) of ev)
                                           time))
                  [for xpnode from 1 to netsize do (COND
                                                     ((EQUAL (fetch (PNODE STATUS) of (ELT NET xpnode))
                                                             "Busy")
                                                       (SETA utilization xpnode (IPLUS (ELT utilization xpnode)
                                                                                       delta]
                  (SETQ time (fetch (EVENT TIME) of ev))
                  (SETQ newtimeflag T))
                (T (SETQ newtimeflag NIL)))
              [COND
                (newtimeflag (COND
                               (display!time!flag (DISPLAY)
                                                  (DISPLAY "Time: " time)
                                                  (DISPLAY)))
                             (COND
                               (display!node!flag (DISPLAY)
                                                  (DISPLAY "-- Node Status --")
                                                  (DISPLAY)
                                                  (for xpnode from 1 to netsize do (DISPLAY!NODE xpnode]
              (COND
                (display!events!flag (DISPLAY!EVENT ev)))
              (SELECTQ (CAR (fetch (EVENT DATA) of ev))
                       [DISPLAY!EVENT (COND
                                        ((EQ (fetch (DISPLAY!EVENT TYPE) of (fetch (EVENT DATA) of ev))
                                             'TASK)
                                          (PROCESS!DISPLAY!EVENT (fetch (EVENT DATA) of ev]
                       [INTERNAL!EVENT (SETQ eventflag (PROCESS!INTERNAL!EVENT (fetch (EVENT DATA) of ev]
                       [MESSAGE (SETQ eventflag T)
                                (SETQ evdata (fetch (EVENT DATA) of ev))
                                (SETQ xaddressee (fetch (MESSAGE ADDRESSEE) of evdata))
```

```
                                    (COND
                                      [(EQUAL xaddressee '(0))
                                        (APPLY final!function (LIST (fetch (MESSAGE ORIGINATOR) of evdata)
                                                                    (fetch (FINAL!REPORT NAME)
                                                                       of (fetch (MESSAGE TEXT) of evdata))
                                                                    (fetch (FINAL!REPORT RESULT!DESCRIPTION)
                                                                       of (fetch (MESSAGE TEXT) of evdata]
                                      (T (SETQ messagecounter (ADD1 messagecounter))
                                         [if (EQUAL evdata:MESSAGE.ADDRESSEE "*")
                                             then (SETQ bdcstcounter (ADD1 bdcstcounter))
                                                  (SETQ nodelist (SORT nodelist 'RANDOMCOMPARE]
                                         (SELECTQ (CAR evdata:MESSAGE.TEXT)
                                                  (TASK!ANNOUNCEMENT (SETQ tacounter (ADD1 tacounter)))
                                                  (BID (SETQ bidcounter (ADD1 bidcounter)))
                                                  (ANNOUNCED!AWARD (SETQ aacounter (ADD1 aacounter)))
                                                  (DIRECTED!AWARD (SETQ dacounter (ADD1 dacounter)))
                                                  [ACKNOWLEDGEMENT (if (EQUAL evdata:MESSAGE.TEXT:ACKNOWLEDGEMENT.RESPONSE
                                                                              'ACCEPTANCE)
                                                                       then (SETQ acccounter (ADD1 acccounter))
                                                                     else (SETQ recounter (ADD1 recounter]
                                                  (INTERIM!REPORT (SETQ ircounter (ADD1 ircounter)))
                                                  (FINAL!REPORT (SETQ frcounter (ADD1 frcounter)))
                                                  (TERMINATION (SETQ tecounter (ADD1 tecounter)))
                                                  (NODE!AVAILABILITY!ANNOUNCEMENT (SETQ nacounter (ADD1 nacounter)))
                                                  (REQUEST (SETQ rqcounter (ADD1 rqcounter)))
                                                  (INFORMATION (SETQ imcounter (ADD1 imcounter)))
                                                  NIL)
                                         (for xpnode in nodelist do (PROCESS!MESSAGE xpnode xaddressee evdata]
                      NIL)
                (COND
                  (eventflag (SETQ rtime time)))
               until (NOT (fetch (EVENT LLINK) of eventlist)))
             (COND
               (display!banners!flag (DISPLAY)
                                     (DISPLAY)
                                     (DISPLAY ":::::::::::::::::::::::::::: End of Simulation ::::::::::::::::::::::::::::::")
                                     (DISPLAY)))
             (COND
               (display!statistics!flag (DISPLAY!STATISTICS]))
----------
```

Calls: DISPLAY!EVENT DISPLAY!NODE DISPLAY!PARAMETERS DISPLAY!STATISTICS INITIALIZE INSTALL!INTERNAL!EVENT NEXT!EVENT PROCESS!DISPLAY!EVENT PROCESS!INTERNAL!EVENT PROCESS!MESSAGE RANDOMCOMPARE STORE!OBJECT STORE!TASK!OBJECT

Called by: CNET

Freevars: NET aacounter acccounter bdcstcounter bidcounter dacounter display!banners!flag display!events!flag display!node!flag display!parameter!flag display!statistics!flag display!time!flag eventcounter eventlist final!function frcounter imcounter initial!function ircounter messagecounter nacounter netsize nodelist recounter rqcounter rtime tacounter tecounter time utilization

Explanation: Performs the main contract net simulation. Initializes the net and calls the initial user function. Sets up contracts as indicated by that function. Then processes events from the event list until no more events remain to be processed. Then displays statistics if required.

```
(STORE!OBJECT
  [LAMBDA (xpnode xobject xinstance)                                              (* rgs: " 8-Sep-78 00:16")
    (PROG (xpnode! kb index otherindex)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ kb (fetch (PNODE KNOWLEDGE!BASE) of xpnode!))
          (COND
            ((MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE!BASE))
              [SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE]
              (SETQ index (CONS xinstance index))
              (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE)
                            'replace index))
            (T [SETQ otherindex (CAR (SOME (fetch (KNOWLEDGE!BASE OTHER) of kb)
                                           (FUNCTION (LAMBDA (x)
                                               (EQUAL (CAR x)
                                                      xobject]
               (COND
                 (otherindex [SETQ otherindex (CONS (CAR otherindex)
                                                    (CONS xinstance (CDR otherindex]
                             (FRPLACA [SOME (fetch (KNOWLEDGE!BASE OTHER) of kb)
                                            (FUNCTION (LAMBDA (x)
                                                (EQUAL (CAR x)
                                                       xobject]
                                      otherindex))
                 (T (replace (KNOWLEDGE!BASE OTHER) of kb with (CONS (CONS xobject (CONS xinstance))
                                                                      (fetch (KNOWLEDGE!BASE OTHER) of kb])
----------
```

Called by: $INITIALIZE INITIALIZE MAKE!BID PROCESS!DIRECTED!AWARD PROCESS!INFORMATION QINITIALIZE SIMULATE STORE!TASK!OBJECT

Freevars: NET

Explanation: Stores an object of type "xobject" in the knowledge base of node "xpnode". "xinstance" is the object. A knowledge base is a record with slots that correspond to the objects recognized by all nodes. Such objects are listed in each slot. A knowledge base also has an 'other' slot used to hold a list of lists of dynamically defined objects. Each such list has a header that corresponds to the type of object.

rgs: 16-Oct-78 17:46 [CNET]

```
(STORE!TASK!OBJECT
  [LAMBDA (xpnode xtype xspecification)                                          (* rgs: "16-Oct-78 17:46")
    (PROG (xpnode! xtask!template xtask)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ xtask!template (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE xtype))
          (SETQ xtask (create TASK NAME ←(fetch (PNODE TASKCOUNTER) of xpnode!)
                              TYPE ← xtype ANNOUNCEMENT!PROCEDURE ←(fetch (TASK!TEMPLATE ANNOUNCEMENT!PROCEDURE)
                                                                          of xtask!template)
                              ANNOUNCEMENT!RANKING!PROCEDURE ←(fetch (TASK!TEMPLATE ANNOUNCEMENT!RANKING!PROCEDURE)
                                                                       of xtask!template)
                              BID!CONSTRUCTION!PROCEDURE ←(fetch (TASK!TEMPLATE BID!CONSTRUCTION!PROCEDURE)
                                                                 of xtask!template)
                              BID!RANKING!PROCEDURE ←(fetch (TASK!TEMPLATE BID!RANKING!PROCEDURE) of xtask!template)
                              AWARD!PROCEDURE ←(fetch (TASK!TEMPLATE AWARD!PROCEDURE) of xtask!template)
                              REFUSAL!PROCEDURE ←(fetch (TASK!TEMPLATE REFUSAL!PROCEDURE) of xtask!template)
                              REFUSAL!PROCESSING!PROCEDURE ←(fetch (TASK!TEMPLATE REFUSAL!PROCESSING!PROCEDURE)
                                                                   of xtask!template)
                              REPORT!ACCEPTANCE!PROCEDURE ←(fetch (TASK!TEMPLATE REPORT!ACCEPTANCE!PROCEDURE)
                                                                  of xtask!template)
                              TERMINATION!PROCEDURE ←(fetch (TASK!TEMPLATE TERMINATION!PROCEDURE) of xtask!template)
                              INFORMATION!ACCEPTANCE!PROCEDURE ←(fetch (TASK!TEMPLATE INFORMATION!ACCEPTANCE!PROCEDURE)
                                                                        of xtask!template)
                              EXECUTION!PROCEDURE ←(fetch (TASK!TEMPLATE EXECUTION!PROCEDURE) of xtask!template)
                              SPECIFICATION ← xspecification))
          (STORE!OBJECT xpnode 'TASK xtask)
          (replace (TASK!TEMPLATE TASKS) of xtask!template with (CONS (fetch (TASK NAME) of xtask)
                                                                      (fetch (TASK!TEMPLATE TASKS) of xtask!template)))
          (replace (PNODE TASKCOUNTER) of xpnode! with (ADD1 (fetch (PNODE TASKCOUNTER) of xpnode!)))
          (RETURN (fetch (TASK NAME) of xtask])
----------
```

Calls: RETRIEVE!OBJECT STORE!OBJECT

Called by: GENERATE!SUBTASK PROCESS!ANNOUNCED!AWARD PROCESS!DIRECTED!AWARD SIMULATE

Freevars: NET

Explanation: Stores a task object of type "xtype" in the knowledge base of node "xpnode". "xspecification" is the 'task!specification'. The 'task!template' for the task type is used to copy pointers to the required procedures for the task. The name of the task is simply the number of tasks that have been generated by "xpnode".
   Returns the task name.

rgs: 16-Sep-78 01:53 [CNET]

```
(SUSPEND
  [LAMBDA NIL                                                                    (* rgs: "16-Sep-78 01:53")
    (AU-REVOIR 'SUSPEND])
----------
```

Called by: EXTEND!BOARD QRECEIVE

Explanation: Does an AU-REVOIR and returns the keyword 'SUSPEND as the possibilities list value. Called directly by user task execution procedures to suspend processing of tasks.

```
rgs:  9-Aug-78 21:08 [CNET]                                                                    TERMINATE
                                                                                             -------------
(TERMINATE
  [LAMBDA NIL                                                          (* rgs: " 9-Aug-78 21:08")
                                                                       (* perhaps should send a report here)
    (ADIEU 'TERMINATE])
----------
Called by: $TEST EXTEND!BOARD QRECEIVE
```

Explanation: Does an ADIEU and returns the keyword 'TERMINATE as the possibilities list value. Called directly by user task execution procedures to terminate processing of tasks.

```
rgs: 18-Oct-78 21:05 [CNET]                                                         TERMINATE!SUBCONTRACTS
                                                                                  --------------------------
(TERMINATE!SUBCONTRACTS
  [LAMBDA (xpnode xcontract xtime)                                     (* rgs: "18-Oct-78 21:05")
    (PROG (tmpsc)
          (COND
            ((NULL xtime)
              (SETQ xtime 0)))
          [for x in (CDR (fetch (CONTRACT SUBCONTRACTS) of xcontract))
             do [COND
                  [(NOT (EQUAL 0 (fetch (SUBCONTRACT CONTRACTOR) of x)))
                    (SENDMESSAGE xpnode (IPLUS time tt xtime)
                                 (fetch (SUBCONTRACT CONTRACTOR) of x)
                                 (create TERMINATION NAME ←(fetch (SUBCONTRACT NAME) of x]
                  (T (NODE!SEARCH xpnode (fetch (SUBCONTRACT NAME) of x)
                                  'ANNOUNCED T)
                     (INSTALL!DISPLAY!EVENT time xpnode 'SIMULATION (APPEND '(Terminated Contract)
                                                                            (fetch (SUBCONTRACT NAME) of x]
                [for y in (fetch (SUBCONTRACT SUCCESSORS) of x) do (SETQ tmpsc (FIND!SUBCONTRACT xpnode y))
                                                                   (replace (SUBCONTRACT PREDECESSORS) of tmpsc
                                                                     with (REMOVE (fetch (SUBCONTRACT NAME) of x)
                                                                                  (fetch (SUBCONTRACT PREDECESSORS)
                                                                                    of tmpsc]
                (for y in (fetch (SUBCONTRACT PREDECESSORS) of x) do (SETQ tmpsc (FIND!SUBCONTRACT xpnode y))
                                                                     (replace (SUBCONTRACT SUCCESSORS) of tmpsc
                                                                       with (REMOVE (fetch (SUBCONTRACT NAME)
                                                                                      of x)
                                                                                    (fetch (SUBCONTRACT SUCCESSORS)
                                                                                      of tmpsc]
          (replace (CONTRACT SUBCONTRACTS) of xcontract with NIL])
----------
Calls:     FIND!SUBCONTRACT INSTALL!DISPLAY!EVENT NODE!SEARCH SENDMESSAGE

Called by: PROCESS!TERMINATION UPDATE!NODE

Freevars:  time tt
```

Explanation: Terminates all outstanding subcontrracts of the contract with name "xname" (held by node "xpnode") at time "xtime". Updates predecessors and successor (some question on this for future).

```
(UPDATE!ACTIVE!TASK!ANNOUNCEMENTS
  [LAMBDA (xpnode)                                                 (* rgs: "16-Oct-78 21:52")
    (PROG (xpnode! active)
          (SETQ xpnode! (ELT NET xpnode))
          (SETQ active (fetch (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!))
          (COND
            (active (replace (PNODE ACTIVE!TASK!ANNOUNCEMENTS) of xpnode!
                       with (for x in active collect x when (IGREATERP (fetch (ACTIVE!TASK!ANNOUNCEMENT EXPIRATION!TIME)
                                                                             of x)
                                                                        time])
```

Called by: NEXT!CONTRACT PROCESS!TASK!ANNOUNCEMENT

Freevars: NET time

Explanation: Deletes all active task announcements in node "xpnode" whose expiration time has passed.

Freevars: NET ntermcs task!time time

Explanation: Continues processing of contract named "xname" in node "xpnode". "xdata" is the pointer to the task execution procedure (as a generator).
If "xdata" is 'TERMINATE, then the contract is terminated (this occurs when the task execution procedure calls the function TERMINATE).
All outstanding subcontracts are terminated.
If the terminated state contains more than 'ntermcs' contracts then the oldest contract is discarded, after presenting it to the 'termination!procedure' for its task (if such a procedure exists).
Processing starts on the next contract in the 'ready' state.
If "xdata" is 'SUSPEND, then the contract is suspended (this occurs when the task execution procedure calls the function SUSPEND).
Processing starts on the next contract in the 'ready' state.
Otherwise, TRYNEXT is executed and another 'node!update' event is placed on the event list.

rgs: 16-Oct-78 21:53 [CNET]

UPDATE!OBJECT

```
(UPDATE!OBJECT
  [LAMBDA (xpnode xobject xkey xslot xvalue)                              (* rgs: "16-Oct-78 21:53")
    (PROG (xpnode! kb index otherindex otherindexl xinstance)
          (COND
            ((MEMBER xslot (RECORDFIELDNAMES xobject))
              (SETQ xpnode! (ELT NET xpnode))
              (SETQ kb (fetch (PNODE KNOWLEDGE!BASE) of xpnode!))
              [COND
                [(MEMBER xobject (RECORDFIELDNAMES 'KNOWLEDGE!BASE))
                  [SETQ index (RECORDACCESS xobject kb (RECLOOK 'KNOWLEDGE!BASE]
                  [SETQ xinstance (CAR (SOME index (FUNCTION (LAMBDA (x)
                                                     (EQUAL (CADR x)
                                                            xkey]
                  (COND
                    (xinstance (RECORDACCESS xslot xinstance (RECLOOK xobject)
                                             'replace xvalue]
                (T (SETQ index (fetch (KNOWLEDGE!BASE OTHER) of kb))
                   [SETQ otherindex (CDAR (SOME index (FUNCTION (LAMBDA (x)
                                                         (EQUAL (CAR x)
                                                                xobject]
                   [SETQ xinstance (CAR (SOME otherindex (FUNCTION (LAMBDA (x)
                                                       (EQUAL (CADR x)
                                                              xkey]
                   (COND
                     (xinstance (RECORDACCESS xslot xinstance (RECLOOK xobject)
                                              'replace xvalue]
              (RETURN xinstance])
----------
```

Freevars: NET

Explanation: Replaces the value of the "xslot" slot of the object of type "xobject" in node "xpnode". "xkey" is the key thatt is matched to find the object, and "xvalue" is the new value for the named slot. "xkey" must be the value of the first slot for the object.

```
(UPDATE!NODE
  [LAMBDA (xpnode xname xdata)                                   (* rgs: "16-Oct-78 21:53")
    (PROG (xpnode! xcontract temp templ xtn xta xtt xtermproc)
          (SETQ temp (CAR xdata))
          (SETQ xpnode! (ELT NET xpnode))
          (COND
            [(SAME!STATUS!CHECK xpnode xname)
              (SETQ xcontract (CAR (fetch (PNODE EXECUTING) of xpnode!)))

          (* if the value returned on the possibilities list is SUSPEND or TERMINATE, then take the associated
          action. Otherwise reschedule the contract through TRYNEXT and place a new node!update event on the
          event list)


              (SELECTQ (CDR xdata)
                       (TERMINATE (replace (CONTRACT STATE) of xcontract with 'TERMINATED)
                                  (replace (PNODE TERMINATED) of xpnode! with (CONS xcontract (fetch (PNODE TERMINATED)
                                                                                                     of xpnode!)))
                                  [COND
                                    ((ILESSP ntermcs (LENGTH (fetch (PNODE TERMINATED) of xpnode!)))
                                      (DREVERSE (fetch (PNODE TERMINATED) of xpnode!))
                                      [SETQ xtn (fetch (CONTRACT TASK) of (CAR (fetch (PNODE TERMINATED) of xpnode!]
                                      (SETQ xta (RETRIEVE!OBJECT xpnode 'TASK xtn))
                                      (SETQ xtermproc (fetch (TASK TERMINATION!PROCEDURE) of xta))
                                      [COND
                                        [xtermproc (APPLY (fetch (PROCEDURE CODE) of (RETRIEVE!OBJECT xpnode
                                                                                                         'PROCEDURE
                                                                                                         xtermproc))
                                                          (LIST xpnode (CAR (fetch (PNODE TERMINATED) of xpnode!]
                                        (T (SETQ xtt (RETRIEVE!OBJECT xpnode 'TASK!TEMPLATE (fetch (TASK TYPE)
                                                                                                    of xta)))
                                           (replace (TASK!TEMPLATE TASKS) of xtt
                                              with (REMOVE xtn (fetch (TASK!TEMPLATE TASKS) of xtt)))
                                           (DELETE!OBJECT xpnode 'TASK xtn)
                                           (DELETE!OBJECT xpnode 'CONTRACT (fetch (CONTRACT NAME)
                                                                             of (CAR (fetch (PNODE TERMINATED)
                                                                                          of xpnode!]
                                      (replace (PNODE TERMINATED) of xpnode!
                                         with (DREVERSE (CDR (fetch (PNODE TERMINATED) of xpnode!]
                                  (INSTALL!DISPLAY!EVENT (IPLUS time task!time)
                                                         xpnode
                                                         'SIMULATION
                                                         (APPEND '(Terminated Contract)
                                                                 xname))
                                  (TERMINATE!SUBCONTRACTS xpnode xcontract)
                                  (NEXT!CONTRACT xpnode 'TERMINATION))
                       (SUSPEND (replace (CONTRACT STATE) of xcontract with 'SUSPENDED)
                                                                             (* must retain the pointer to the
                                                                             possibilities list)
                                (replace (PNODE SUSPENDED) of xpnode! with (CONS (CONS xcontract temp)
                                                                                  (fetch (PNODE SUSPENDED) of xpnode!)))
                                (INSTALL!DISPLAY!EVENT (IPLUS time task!time)
                                                       xpnode
                                                       'SIMULATION
                                                       (APPEND '(Suspended Contract)
                                                               xname))
                                (NEXT!CONTRACT xpnode 'TERMINATION))
                       (PROGN (SETQ templ (RESUME!TASK temp))
                              (INSTALL!INTERNAL!EVENT (IPLUS time task!time)
                                                      xpnode xname 'NODE!UPDATE (CONS temp templ]
            (T (RELEASE!TASK temp])
----------
Calls:     DELETE!OBJECT INSTALL!DISPLAY!EVENT INSTALL!INTERNAL!EVENT NEXT!CONTRACT RELEASE!TASK RESUME!TASK
           RETRIEVE!OBJECT SAME!STATUS!CHECK TERMINATE!SUBCONTRACTS

Called by: PROCESS!INTERNAL!EVENT
```

```
rgs: 10-Aug-78 15:35 [CNET]                                                   UPDATE!TASK!TIME
                                                                            ---------------------
(UPDATE!TASK!TIME
  [LAMBDA (t)                                                          (* rgs: "10-Aug-78 15:35")
    (SETQ task!time (IPLUS task!time (ITIMES gain (COND
                                                   ((IGREATERP t -1)
                                                     t)
                                                   (T 0])
----------
Called by: $TEST EXTEND!BOARD

Freevars:  gain task!time

Explanation:  Updates the 'task!time' for the calling task by 'gain' times "t" units.



rgs: 17-Oct-78 22:14 [CNET]                                                             VALUEP
                                                                                    ----------
(VALUEP
  [LAMBDA (xvalue)                                                     (* rgs: "17-Oct-78 22:14")
    (COND
      ((EQ (CAR xvalue)
           'QUOTE)
        (CADR xvalue))
      (T NIL])
----------
```

```
December 9, 1978 5:56PM in <VANMELLE>WW.SAV;81121 by RGSMITH


Fns on INTERPRET:

    INTERPRETBLOCK                    INTERPP                        INTERPU
      INTERPRET                       INTERPQ                        INTERPV
      BINDSYMBOL                      INTERPT                        QUERY

Block INTERPRETBLOCK
    Entries:    INTERPRET
    Internal:   BINDSYMBOL, INTERPP, INTERPQ, INTERPT, INTERPU, INTERPV, QUERY
    Specvars:   PHRASE, CLASSES, QR, ELLIPSISFLG, BINDINGS, FUNCTIONS, POSSIBLEGRAMMARS, ORIGINALPHRASE, TOPFLG,
                TEMPLATE, REMTEMPLATE
```

```
(INTERPRET
  [LAMBDA (PHRASE GRAMMAR CLASSES BINDINGS NOELLIPSIS NOFAILRECORDS)            (* rgs: "25-Oct-78 00:56")
    (PROG (QR POSSIBLEGRAMMARS (ORIGINALPHRASE PHRASE))
          [COND
            [(fetch (QRESULTS QMATCH) of (SETQ QR (QUERY PHRASE GRAMMAR BINDINGS NIL T]         |
            ((AND (NOT NOELLIPSIS)
                  BINDINGS)
              (WRITE "Trying ellipsis...")
              (SETQ QR (QUERY PHRASE GRAMMAR BINDINGS T T]
          (RETURN (create INTERPRETATION MATCH ←(fetch (QRESULTS QMATCH) of QR)                 |
                          RESULTS ←(COND
                            ((fetch (QRESULTS QMATCH) of QR)                                    |
                              (fetch (QRESULTS VALUE) of QR))                                   |
                            (T POSSIBLEGRAMMARS))
                          REMAININGPHRASE ←(fetch (QRESULTS RP) of QR)                          |
                          BOUNDCLASSES ←(fetch (QRESULTS BINDINGS) of QR])
----------
```

Calls: QUERY

Called by: CILPARSE

Explanation: A semantic grammar parser, modeled after Hendrix's LIFER. See also A. Bonnet, "BAOBAB, A Parser for a Rule-based System Using a Semantic Grammar, STAN-78-668 (HPP-78-10), Dept. of Computer Science, Stanford University, June 1978. This is a general explanation of the operation of INTERPRET and its associated functions.

"PHRASE" is the phrase to be parsed. It is converted to upper case before the parse is attempted. "GRAMMAR" is the top level grammar. "CLASSES" is a list of CLASSES in the top level grammar. "BINDINGS" is an optional list of initial bindings that is used by INTERPRET to perform ellipstic resolution. If "NOELLIPSIS" is NIL and "BINDINGS" is non-NIL then "BINDINGS" is used to perform elliptic resolution. Otherwise only one parse is attempted. In either case, pronomial reference is resolved. If "NOFAILRECORDS" is T then no FAILURE records are returned if the parse fails.

INTERPRET returns an INTERPRETATION record with the following fields:

MATCH: T if the phrase has been parsed; else NIL.
RESULTS: the results of applying the ACTION!FUNCTIONs (see below) if the phrase has been parsed; else NIL if no TEMPLATEs (see below) have been partially matched; else a list of FAILURE records if some TEMPLATEs have been partially matched. Currently, such a list can contain duplicate records. FAILURE records are only returned for top-level templates. A FAILURE record has the following fields:
TEMPLATE: the TEMPLATE that was partially matched.
FUNCTION: the list of (SEMANTIC!PREDICATE ACTION!FUNCTION) for the TEMPLATE that was partially matched.
REMTEMPLATE: the remaining portion of the TEMPLATE that was not matched.
REMPHRASE: the remaining portion of the phrase that was not matched.
FBINDINGS: the bindings for the CLASSes up to the point where the failure occurred.
REMAININGPHRASE: if a parse has been completed, then a list of the remaining words in the phrase (if any) that could not be parsed; else the complete phrase.
BOUNDCLASSES: the bindings for the top-level CLASSes in the grammar; i.e., only the bindings for CLASSes mentioned explicitly in the top-level grammar. Bindings for CLASSes in subgrammars (see below) are not returned. NIL if the parse failed.

The following notes specify the form of a grammar for use by INTERPRET. They also give more detail on how INTERPRET operates.

GRAMMAR: A list of TEMPLATEs (or sequences of TEMPLATEs), SEMANTIC!PREDICATEs and ACTION!FUNCTIONs. If the TEMPLATE can be matched syntactically, then the SEMANTIC!PREDICATE is evaluated using the bindings, ((CLASS value)...), as an a-list. If it returns T then the ACTION!FUNCTION is evaluated using the bindings as an a-list. T is a valid SEMANTIC!PREDICATE. If a sequence of TEMPLATEs is used, then the SEMANTIC!PREDICATE and ACTION!FUNCTION are evaluated in the context of the bindings for the first TEMPLATE in the sequence that is successfully matched.

```
( (TEMPLATE-1 TEMPLATE-2 ... (SEMANTIC!PREDICATE-1 ACTION!FUNCTION-1))
(TEMPLATE-3 (SEMANTIC!PREDICATE-2 ACTION!FUNCTION-2) )
```

TEMPLATE: A list of CLASSes and/or ANCHOR!WORDs (i.e., words that are to be literally matched). (Sequences of) CLASSes or ANCHOR!WORDs enclosed in parentheses in a TEMPLATE are optional. This is done by first matching all such optional sequences, then proceeding with an attempted match on the TEMPLATE. If optional sequences cannot be matched, but the remainder of the TEMPLATE can be matched, then the parse will still succeed. If they can be matched, then the matching is done in the usual way. (This is also the way in which recursive subgrammars are handled. For example: noun-phrase ← adjective (noun-phrase)

```
(CLASS-1 CLASS-2 (CLASS-3) ANCHOR!WORD-1 ... CLASS-N)
```

Each CLASS has one of the following properties:

PRONOUNS: If a CLASS has the PRONOUN property then, when it is matched (see POSSIBLEVALUES below), the previous "bindings" of the CLASS are returned; else the TEMPLATE match fails. If the value of PRONOUNS is atomic, then it is treated as a function, and evaluated in the context of the current bindings to get the list of possible values.

POSSIBLEVALUES: a list of values that a member of the CLASS can assume (e.g., this could be part of a dictionary). The value of POSSIBLEVALUES has the following form: (PV-1 PV-2 ... PV-N). If the value of POSSIBLEVALUES is atomic, then it is treated as a function, and evaluated in the context of the current bindings to get a list of this form. Each of the PV-i is used as a possible value for the CLASS. Each PV-i can be one of the following:

1) ATOM. If the ATOM is matched then it is returned as the binding for the match.
2) (ATOM-1 ATOM-2 ...). If the ATOM list is matched then it is returned as the binding for the match.
3) ((ATOM-1 ATOM-2 ...) (ATOM-3 ATOM-4 ...) (...) VALUE). If any of the ATOM lists is matched then VALUE is returned as the binding for the match. VALUE is not evaluated.

PREDICATE: a predicate to be applied to test for class membership. If the predicate is atomic, then it gets one word to test, and is expected to return T or NIL. If it is enclosed in parentheses, then it gets the rest of the phrase, and is expected to return a PREDICATE record. Such a record has the following fields:

REST: the portion of the phrase that remains to be matched after the predicate has been applied.
PVALUE: the result of applying the predicate--either T or NIL.

KLEENE: the CLASS will match anything up to the next TEMPLATE element.

GRAMMARS: the class can itself be a subgrammar in the same format as the top-level grammar. The KLEENE property cannot be used in a subgrammar--stack overflow is the result.

jsb: 13-JAN-78 08:47 [INTERPRET: internal to INTERPRETBLOCK] **BINDSYMBOL**

```
(BINDSYMBOL
  [LAMBDA (CLASS VALUE)                                              (* jsb: "13-JAN-78 08:47")
    (PROG ((ENTRY (FASSOC CLASS BINDINGS)))
          [COND
            (ENTRY (SETQ BINDINGS (for BINDING in BINDINGS unless (EQ BINDING ENTRY) collect BINDING]
          (SETQ BINDINGS (CONS (CONS CLASS VALUE)
                               BINDINGS))
          (RETURN BINDINGS])
----------
```

Called by: INTERPP INTERPT

Freevars: BINDINGS

Explanation: Actually binds a symbol (a word or sequence of words or... -- see INTERPRET) to a CLASS. "CLASS" is the class. "VALUE" is the symbol to be bound to the CLASS. BINDINGS is altered by this function.

```
(INTERPP
  [LAMBDA NIL                                                           (* rgs: "25-Oct-78 01:03")
    (for old REMTEMPLATE on TEMPLATE as GSYMBOL is (CAR REMTEMPLATE) bind QR
       do (COND
            [PHRASE (COND
                      [(LITATOM GSYMBOL)
                        (COND
                          [(FMEMB GSYMBOL CLASSES)                      (* a class definition)
                            (COND
                              [(SETQ QR (GETP GSYMBOL 'KLEENE))         (* special CLASS, matches everything     |
                                                                        until the next GSYMBOL is accepted.)
                                (COND
                                  ((EVAL QR)
                                    [COND
                                      ((NOT (SETQ REMTEMPLATE (CDR REMTEMPLATE)))
                                        (BINDSYMBOL GSYMBOL PHRASE)
                                        (SETQ PHRASE NIL))
                                      ((for XPHRASE on PHRASE when (fetch (QRESULTS QMATCH)                      |
                                                                      of (SETQ QR (INTERPT XPHRASE REMTEMPLATE
                                                                                           '(T NIL)
                                                                                           BINDINGS)))
                                          do (SETQ BINDINGS (fetch (QRESULTS BINDINGS) of QR))                  |
                                             (BINDSYMBOL GSYMBOL (LDIFF PHRASE XPHRASE))
                                             (RETURN T))
                                        (SETQ REMTEMPLATE NIL)
                                        (SETQ PHRASE (fetch (QRESULTS RP) of QR]                               |
                                    (RETURN))
                                  (T (BINDSYMBOL GSYMBOL]
                              [(INTERPV (GETP GSYMBOL 'PRONOUNS))
                                (COND
                                  ((NOT (FASSOC GSYMBOL BINDINGS))
                                    (RETURN]
                              ((INTERPV (GETP GSYMBOL 'POSSIBLEVALUES))
                                (BINDSYMBOL GSYMBOL QR))
                              ((INTERPQ (GETP GSYMBOL 'PREDICATE))       (* simple predicate)
                                (BINDSYMBOL GSYMBOL QR))
                              ((fetch (QRESULTS QMATCH) of (SETQ QR (QUERY PHRASE (GETP GSYMBOL 'GRAMMARS)     |
                                                                          NIL ELLIPSISFLG)))
                                                                        (* a sub-grammar matched)
                                (SETQ PHRASE (fetch (QRESULTS RP) of QR))                                      |
                                (BINDSYMBOL GSYMBOL (fetch (QRESULTS VALUE) of QR)))                           |
                              ((NOT (AND ELLIPSISFLG (FASSOC GSYMBOL BINDINGS)))
                                                                        (* allowing ellipsis and it doesn't     |
                                                                        occur)
                                (RETURN]
                          ((NEQ GSYMBOL (CAR PHRASE))
                            (RETURN))
                          (T (SETQ PHRASE (CDR PHRASE]
                      ((fetch (QRESULTS QMATCH) of (SETQ QR (INTERPT PHRASE GSYMBOL '(T NIL)                  |
                                                                     BINDINGS)))
                        (SETQ PHRASE (fetch (QRESULTS RP) of QR))                                              |
                        (SETQ BINDINGS (fetch (QRESULTS BINDINGS) of QR]                                       |
            ((AND (LITATOM GSYMBOL)
                  (FMEMB GSYMBOL CLASSES))
              (COND
                ((NOT (AND ELLIPSISFLG (FASSOC GSYMBOL BINDINGS)))
                  (RETURN])
```

Calls: BINDSYMBOL INTERPQ INTERPT INTERPV QUERY

Called by: INTERPT

Freevars: BINDINGS CLASSES ELLIPSISFLG PHRASE REMTEMPLATE TEMPLATE

Explanation: Tries to match a template against a phrase. "PHRASE" is the phrase to be matched. "TEMPLATE" is the template to be used. "BINDINGS" is the list of current bindings. Returns a QRESULTS record with the following field settings:

```
rgs: 25-Oct-78 01:06 [INTERPRET: internal to INTERPRETBLOCK]                                INTERPQ
                                                                                          -----------
(INTERPQ
  [LAMBDA (PREDICATE)                                                    (* rgs: "25-Oct-78 01:06")
    (COND
      (PREDICATE                                                         (* there is a predicate)
             (COND
               ((LITATOM PREDICATE)
                 (COND
                   ((SETQ QR (APPLY* PREDICATE (CAR PHRASE)))            (* A SIMPLE PREDICATE)
                     (SETQ PHRASE (CDR PHRASE))
                     QR)))
               ((SETQ QR (APPLY* (CAR PREDICATE)
                                 PHRASE))
                 (SETQ PHRASE (fetch (PREDICATE REST) of QR))
                 (SETQ QR (fetch (PREDICATE PVALUE) of QR])
----------
Called by: INTERPP

Freevars:  PHRASE QR

Explanation:  Applies a predicate to the remaining phrase (see INTERPRET for more detail). "PREDICATE" is the
           predicate to be applied.



rgs: 25-Oct-78 01:08 [INTERPRET: internal to INTERPRETBLOCK]                                INTERPT
                                                                                          -----------
(INTERPT
  [LAMBDA (PHRASE TEMPLATE FUNCTIONS BINDINGS TOPFLG)                    (* rgs: "25-Oct-78 01:08")
    (PROG (REMTEMPLATE)
          (for ENTRY in TEMPLATE when (NOT (LITATOM ENTRY)) do (for GSYMBOL in ENTRY
                                                                  when (AND (FMEMB GSYMBOL CLASSES)
                                                                            (NOT (FASSOC GSYMBOL BINDINGS)))
                                                                  do (BINDSYMBOL GSYMBOL)))
          (INTERPP)
          (COND
            [(AND (NULL REMTEMPLATE)
                  (OR (NOT TOPFLG)
                      (NEQ ORIGINALPHRASE PHRASE))
                  (EVALA (fetch (FUNCTIONS SEMANTIC!PREDICATE) of FUNCTIONS)
                         BINDINGS))                                      (* it worked!)
              (RETURN (create QRESULTS QMATCH ← T RP ← PHRASE BINDINGS ← BINDINGS VALUE ←(EVALA
                               (fetch (FUNCTIONS ACTION!FUNCTION) of FUNCTIONS)
                               BINDINGS]
            ((AND TOPFLG (NOT NOFAILRECORDS)
                  (NOT ELLIPSISFLG)
                  (NEQ REMTEMPLATE TEMPLATE))
              (SETQ POSSIBLEGRAMMARS
                (CONS (create FAILURE TEMPLATE ← TEMPLATE FUNCTIONS ← FUNCTIONS REMPHRASE ← PHRASE REMTEMPLATE ←
                              REMTEMPLATE FBINDINGS ← BINDINGS)
                      POSSIBLEGRAMMARS])
----------
Calls:     BINDSYMBOL INTERPP

Called by: INTERPP QUERY

Freevars:  CLASSES ELLIPSISFLG NOFAILRECORDS ORIGINALPHRASE POSSIBLEGRAMMARS

Explanation:  Tries to match an element of a grammar against a phrase. If the match is successful, then the
           associated SEMANTIC!PREDICATE is evaluated.  If that is successful, then the associated ACTION!FUNCTION is
           evaluated.  "PHRASE" is the phrase to be matched. "TEMPLATE" is the element of the grammar (see INTERPRET
           for more detail). "BINDINGS" is the current list of bindings.
```

```
(INTERPU
  [LAMBDA (PATTERN)
    (PROG [(REMPHRASE (FNTH PHRASE (FLENGTH PATTERN]
          (COND
            ((AND REMPHRASE (for PWORD in PATTERN as WORD in PHRASE always (EQ PWORD WORD)))
              (SETQ PHRASE (CDR REMPHRASE))
              (RETURN T])
```

Called by: INTERPV

Freevars: PHRASE

Explanation: Returns T if the N words in "PATTERN" match the first N words in "PHRASE". Also removes the words from "PHRASE" when a match is found.

```
(INTERPV
  [LAMBDA (VALUES)                                                  (* bvm: "19-Feb-78 16:39")
    [COND
      ((AND VALUES (ATOM VALUES))
        (SETQ VALUES (EVALA (LIST VALUES)
                            BINDINGS]
    (for ENTRY in VALUES do (SETQ QR ENTRY)
                            (COND
                              [(ATOM ENTRY)
                                (COND
                                  ((EQ ENTRY (CAR PHRASE))
                                    (SETQ PHRASE (CDR PHRASE))
                                    (RETURN T]
                              [(LISTP (CAR ENTRY))
                                (SETQ QR (CAR (FLAST ENTRY)))
                                (COND
                                  ((for PATTERN in ENTRY unless (EQ QR PATTERN) thereis (INTERPU PATTERN))
                                    (RETURN T]
                              ((INTERPU ENTRY)
                                (RETURN T])
```

Calls: INTERPU

Called by: INTERPP

Freevars: BINDINGS PHRASE QR

Explanation: Tries to match a pronoun or possible value against the phrase. "VALUES" is the pronoun or possible value to be matched (see INTERPRET for more detail).

```
(QUERY
  [LAMBDA (PHRASE GRAMMAR BINDINGS ELLIPSISFLG TOPFLG)                        (* rgs: "25-Oct-78 01:11")
    (PROG (QR)
          [COND
            (PHRASE (for ENTRY in GRAMMAR unless (fetch (QRESULTS QMATCH) of QR) bind FUNCTIONS
                       do (SETQ FUNCTIONS (CAR (FLAST ENTRY)))
                          (for TEMPLATE in ENTRY unless (EQ FUNCTIONS TEMPLATE)
                             thereis (SETQ QR (INTERPT PHRASE TEMPLATE FUNCTIONS BINDINGS TOPFLG]
          (RETURN QR])
----------
```

Calls: INTERPT

Called by: INTERPP INTERPRET

Explanation: Actually carries out a parse attempt for the phrase. "PHRASE" is the phrase to be parsed. "GRAMMAR" is the grammar to be used for the parse. "BINDINGS" is the list of current bindings. "ELLIPSISFLG" is T if elliptical reference is to be resolved. "TOPFLG" is T if QUERY is working on the top-level grammar. Returns T if a parse has been successfully completed; else returns a QRESULTS record with the RI field set to PHRASE.

```
December 9, 1978 5:58PM in <VANMELLE>WW.SAV;81121 by RGSMITH


Fns on UTILITY:

    ASKFORFILENAME       DISPLAY              PRINTPROP&VAL       SPRINTCOUNT         UGETHASHFILE
    ASKFORFUNCTIONNAME   DISPLAYHELP          PRINTRECORD         SPRINTPUNC          WRITE
    ASKFORNUMBER         GETFILE              SPRINT              SPRINTSEPR          WRITE1
    ASKFORYESNO          INFILEDIR            SPRINT1             SPRINTSTRING        WRITE3
    ASKYESNO             OPENHASHFILEVARS     SPRINTATOM          TTYOUT              WRITEARG
    CTRLO.NLSETQ         OPENMYCINHASHFILE
```

```
rgs: 13-Oct-78 01:05 [UTILITY]                                                             ASKFORFILENAME
                                                                                         ------------------

(ASKFORFILENAME
  [LAMBDA (xmode xdefault)                                                    (* rgs: "13-Oct-78 01:05")
    (PROG (tempjfn tempfile)
          [COND
            (xdefault (SETQ xdefault (SELECTQ xmode
                                              (INPUT (INFILEP xdefault))
                                              (OUTPUT (OUTFILEP xdefault))
                                              NIL]
      LOOP(WRITE1 "File Name for " xmode " ")
          (COND
            (xdefault (WRITE1 "[" xdefault "] ** "))
            (T (WRITE1 "** ")))
          (SETQ tempjfn (RESETLST (RESETSAVE (INTERRUPTCHAR 4))
                                  (RESETSAVE (INTERRUPTCHAR 5))
                                  (JSYS 16 (SELECTQ xmode
                                                    (INPUT 15033171968)
                                                    -10736631808)
                                        16777281)))
          (SETQ tempfile (JFNS tempjfn))
          (COND
            (tempfile (SETQ xdefault tempfile)))
          (COND
            ((NOT (AND (OR tempfile (EQP tempjfn 196685))
                       xdefault))
              (OR (ZEROP (POSITION))
                  (WRITE))
              (WRITE1 (OR (ERSTR tempjfn)
                          "bad response. try again."))
              (WRITE)
              (GO LOOP)))
          (COND
            ((NOT tempfile)
              (WRITE)))
          (RETURN xdefault])
----------
Called by: SET!PARAMETERS
```

Explanation: Asks for a filename. xmode is the mode to be used (READ or WRITE). xdefault is the default, which is returned if <cr> is the response (in the case of WRITE mode a new version is created). Full TENEX recognition is in effect.

```
rgs: 11-Aug-78 23:42 [UTILITY]                                                         ASKFORFUNCTIONNAME
                                                                                     ----------------------

(ASKFORFUNCTIONNAME
  [LAMBDA (xprompt xdefault xhelp)                                            (* rgs: "11-Aug-78 23:42")
    (PROG (temp)
          (SETQ temp T)
          (do (SETQ temp (CAR (TTYIN (LIST xprompt " [" xdefault "] ** ")
                                     NIL xhelp)))
             until (OR (NULL temp)
                       (GETD temp)))
          (COND
            (temp (RETURN temp))
            (T (RETURN xdefault])
----------
Called by: SET!PARAMETERS
```

Explanation: Asks for the name of a function, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. The response is only accepted if it is the name of a function.

```
-------------------- [UTILITY]                                                     ASKFORNUMBER
                                                                                   ----------------

(ASKFORNUMBER
  [LAMBDA (xprompt xdefault xhelp xlb xub)
    (PROG (temp)
          (SETQ temp T)
          [do (SETQ temp (CAR (TTYIN (LIST xprompt " [" xdefault "] ** ")
                                      NIL xhelp)))
             until (OR (NULL temp)
                       (AND (NUMBERP temp)
                            (IGREATERP temp xlb)
                            (COND
                              (xub (ILESSP temp xub))
                              (T T]
          (COND
            (temp (RETURN temp))
            (T (RETURN xdefault])
----------
Called by: QSET!PARAMETERS SET!PARAMETERS
```

Explanation: Asks for a number, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. The response must be greater than xlb and less than xub.

```
rgs: 13-Oct-78 01:06 [UTILITY]                                                      ASKFORYESNO
                                                                                    ---------------

(ASKFORYESNO
  [LAMBDA (xprompt xdefault xhelp)                                  (* rgs: "13-Oct-78 01:06")
    (PROG (temp)
          [SETQ temp (CAR (TTYIN (LIST xprompt " [" (COND
                                                     (xdefault 'YES)
                                                     (T 'NO))
                                                   "] ** ")
                                 (LIST '(Yes . Y)
                                       '(No . N))
                                 xhelp
                                 (LIST 'FIX]
          (RETURN (COND
                    ((NULL temp)
                      xdefault)
                    ((OR (EQ temp 'Y)
                         (EQ temp 'YES))
                      T)
                    (T NIL])
----------
Called by: RESIMULATE SET!PARAMETERS
```

Explanation: Returns T for an affirmative response, using TTYIN. xprompt is the prompt that is displayed. xdefault is the default, which is returned if <cr> is the response. xhelp is the key to a hashfile entry. Essentially like ASKYESNO with a default.

```
(ASKYESNO
  [LAMBDA (QUESTION PROMPTTYPE DEFAULT HELP)                            (* rgs: "13-Oct-78 01:15")
    (SELECTQ PROMPTTYPE
             [(NIL CONFIRM)                                              (* Hacker-type prompts)
               (RESETFORM (SETTERMTABLE ASKUSERTTBL)
                          (PROG (ANSWER BUFS (TYPEAHEAD (READP T)))
                                (COND
                                  [QUESTION (COND
                                              ((LITATOM QUESTION)
                                                (PRIN1 QUESTION T)
                                                (SETQ QUESTION " ? "]
                                  ((EQ PROMPTTYPE 'CONFIRM)
                                    (SETQ QUESTION " [confirm] ")))
                            TOP [COND
                                  ((LISTP QUESTION)
                                    (MAPRINT QUESTION T)
                                    (OR (EQ PROMPTTYPE 'CONFIRM)
                                        (PRIN1 " ? " T)))
                                  (QUESTION (PRIN1 QUESTION T)
                                            (COND
                                              ((NEQ (NTHCHAR QUESTION -1)
                                                    '% )
                                                (SPACES 1 T]
                            READ:                                        (* Do a PBIN to get next character)
                                (SELECTQ (JSYS 59)
                                         ((89 121)                       (* Y)
                                           (PRIN1 "Yes" T)
                                           (SETQ ANSWER T)
                                           (GO DONE:))
                                         ((78 110)                       (* N)
                                           (PRIN1 "No" T)
                                           (GO DONE:))
                                         [(31 15)                        (* crlf)
                                           (COND
                                             ((EQ PROMPTTYPE 'CONFIRM)
                                               (SETQ ANSWER T)
                                               (GO DONE:]
                                         [63                             (* ?)
                                             (TERPRI T)
                                             (SETQ TYPEAHEAD)
                                             (COND
                                               (HELP (COND
                                                       ((LITATOM HELP)
                                                         (DISPLAYHELP HELP))
                                                       (T (SPRINTT HELP)))
                                                     (GO TOP))
                                               (T (PRIN1 (COND
                                                           ((EQ PROMPTTYPE 'CONFIRM)
                                                             "[type carriage return to confirm] ")
                                                           (T "Type Yes or No: "))
                                                         T)
                                                  (GO READ:]
                                         [(127 24)                       (* delete, ↑X to disconfirm)
                                           (COND
                                             ((EQ PROMPTTYPE 'CONFIRM)
                                               (PRIN1 "xxx" T)
                                               (DISMISS 500)
                                               (CLEARBUF T)
                                               (GO DONE:]
                                         NIL)
                                (JSYS 60 7)                              (* Ring terminal bell for inappropriate
                                                                         response)
                                (COND
                                  (TYPEAHEAD                             (* User may have typed ahead before
                                                                         QUESTION was printed.
                                                                         Save buffers and gotry again)
                                              (JSYS 34 64)               (* BKJFN puts back character the PBIN
                                                                      read above)
                                             (DOBE)
                                             (DISMISS 1000)
                                             (SETQ BUFS (CLBUFS))
                                             (SETQ TYPEAHEAD)))
                                    (GO READ:)
                                 DONE:
                                    (COND
                                      (BUFS (BKBUFS BUFS)))
                                    (TERPRI T)
                                    (RETURN ANSWER]
                      (PROGN (COND
                                 (QUESTION (SPRINT QUESTION)))
                           (OR (EQ PROMPTYPE 'NOTERPRI)
                              (TERPRI))
                           (COND
                             (BATCHFLG (WRITE "** ..." (COND
                                                  (DEFAULT "yes")
                                                  (T "no")))
                                        DEFAULT)
                             (T (do (SELECTQ (CAR (TTYIN (SELECTQ PROMPTYPE
                                                              ((T NOTERPRI)
                                                                NIL)
                                                              PROMPTYPE)
                                                       '(YES NO)
                                                       HELP))
                                        (YES (RETURN T))
                                        (NO (RETURN))
                                        (WRITE "Yes or No, please."])
----------
```

Calls: DISPLAYHELP

Called by: OPENMYCINHASHFILE

Globalvars: BATCHFLG

Freevars: ASKUSERTTBL

Explanation: Returns T if the response to QUESTION is affirmative. There are two basic modes: immediate (for hacking-type questions) and standard (using TTYIN); which one depends on the value of PROMPTYPE:

NIL -- (immediate) function behaves roughly like ASKUSER with a yes/no keylst and typeahead permitted (rings bell for incorrect response (not Y or N), clears and saves typeahead if typeahead looks bogus). If QUESTION is a literal atom, it is printed, followed by a "?"; if a list, it is MAPRINTed.

CONFIRM -- like above, except <crlf> is accepted (even expected) as the affirmative response, and <del> or ↑X disconfirm. If QUESTION is NIL, supplies "[confirm]".

T -- (standard) SPRINTTs QUESTION and then calls TTYIN for the standard ** prompt.

NOTERPRI -- like T, but does not print crlf before **.

<other> -- any other prompt is passed to TTYIN.

Additionally: if HELP is specified, it is given if user types a "?" (same as TTYIN's HELP arg). If BATCHFLG is set (i.e. user input is not being taken), DEFAULT (T or NIL) is the response supplied for the non-immediate types.

```
bvm: 16-FEB-77 17:34 [UTILITY] [compiler macro]                                        CTRLO.NLSETQ
                                                                                     ----------------
(CTRLO.NLSETQ
  [NLAMBDA (NLSETX NLSETY)
    (DECLARE (LOCALVARS . T)
             (SPECVARS CTRLO!))                                  (* bvm: "16-FEB-77 17:34")
    (RESETLST (PROG (MACROX (CTRLO! CTRLO!))
                    (COND
                      ((NOT CTRLO!)
                        (RESETSAVE (INTERRUPTCHAR 15 '(CTRLO!)
                                                  T))             (* Only turn on the interrupt if it
                                                                 isn't already)
                        (SETQ CTRLO! T)))
                LP  (SETQ MACROX (ERRORSET NLSETX))
                    (COND
                      ((AND NLSETY (NOT MACROX))                 (* loop until the body exits without
                                                                 error)
                        (GO LP)))
                    (RETURN MACROX])
----------
Called by: DISPLAYHELP

Globalvars: CTRLO!

Explanation:  Evaluates NLSETX under errorset protection, like NLSETQ.  In addition, the ↑O interrupt is armed
           inside here, so that the user may abort with it.  If the second argument (NLSETY) is true and a ↑O happens
           during the evaluation of NLSETX, it is reevaluated, i.e. a ↑O causes the function to loop, and the
           CTRLO.NLSETQ will only exit without error.  The variable CTRLO! is bound to T inside here as a cheap flag
           to indicate that ↑O is on.


rgs:  8-Jul-78 08:51 [UTILITY]                                                              DISPLAY
                                                                                          -----------
(DISPLAY
  [LAMBDA N                                                      (* rgs: " 8-Jul-78 08:51")
    (COND
      (fileflag (for I from 1 to N do (WRITEARG (ARG N I)))
                (TERPRI)))
    (COND
      (termflag (for I from 1 to N do (WRITEARG (ARG N I)
                                                T))
                (TERPRI T])
----------
Calls:     WRITEARG

Called by: DISPLAY!CONTRACT DISPLAY!EVENT DISPLAY!MESSAGE DISPLAY!NODE DISPLAY!PARAMETERS DISPLAY!STATISTICS
           PROCESS!DISPLAY!EVENT QFINALIZE SIMULATE

Freevars:  fileflag termflag

Explanation:  Like WRITE, but writes to the primary output file if fileflag is set, and writes to the terminal if
           termflag is set. If both are set, then writes to both places.
```

```
(DISPLAYHELP
  [LAMBDA (KEY QUIET)                                                     (* bvm: " 7-Mar-78 23:05")
    (PROG (RESULT)
          (RETURN (OR [NOT (SETQ RESULT (CTRLO.NLSETQ (UGETHASHFILE 'HELPFILE KEY NIL NIL
                                                                    (OR QUIET "Helpfile unavailable."]
                      (CAR RESULT])
```

Calls: CTRLO.NLSETQ UGETHASHFILE

Called by: ASKYESNO

Explanation: Copies to primary output the help blurb indexed by KEY. If QUIET is set, will not complain if the hashfile is unavailable (not found or won't open). Returns NIL if no entry found for KEY (and hence nothing was printed); T if the entry was found, or user typed ↑O.

```
(GETFILE
  [LAMBDA (FILE ASK SHOW)                                              (* bvm: " 7-Jun-78 23:41")
    (PROG (FOUND ENTRY)
          [COND
            ((SETQ FOUND (OR (AND (SETQ ENTRY (FASSOC FILE PREFERREDFILES))
                                  (INFILEP (CDR ENTRY)))
                             (INFILEP FILE)))
              (RETURN FOUND))
            ([SETQ ENTRY (FASSOC FILE (OR (LISTP (EVALV 'GETFILELST))
                                          (SETATOMVAL 'GETFILELST]
              (COND
                ((NULL (CADR ENTRY))                                   (* Means forget it)
                  (RETURN))
                ([SETQ FOUND (INFILEP (OR (CDDR ENTRY)
                                          (CDR (FRPLACD (CDR ENTRY)
                                                        (MKATOM (SUBSTRING (CADR ENTRY)
                                                                           1
                                                                           (STRPOS '; (CADR ENTRY]

          (* this gets the latest version, even if that differs from the one we last found.
          We could have stored the MKATOM in the first place, but that would create a possibly superfluous
          atom)


                  (RETURN FOUND))
                (T (DREMOVE ENTRY GETFILELST]
          (COND
            ((SETQ FOUND (for DIR in OTHERDIRS any (INFILEDIR DIR FILE)))
              (COND
                (SHOW (TTYOUT "...from " FOUND)))
              (GO FOUND:))
            ((NOT ASK)
              (RETURN)))
      RD  (COND
            ([AND [SETQ FOUND (CAR (TTYIN (LIST "Directory for" FILE "(or <cr>): ")
                                          NIL
                                          'GETFILE]
                  (NOT (SETQ FOUND (INFILEDIR FOUND FILE]
              (TTYOUT "not found")
              (GO RD)))
      FOUND:
          (SETQ GETFILELST (CONS (LIST FILE FOUND)
                                 GETFILELST))                          (* Save where we found this, in case we
                                                                       have to look again)
          (COND
            ((SETQ ENTRY (FNTH GETFILELST 15))                         (* Drop off old entries)
              (FRPLACD ENTRY)))
          (RETURN FOUND])
```

Calls: INFILEDIR

Called by: OPENMYCINHASHFILE

Globalvars: OTHERDIRS PREFERREDFILES

Freevars: GETFILELST

Explanation: Locates FILE, looking first on the connected directory, then on OTHERDIRS, then if ASK is set asks the user for help. Returns the complete file name of the first file (if any) found which is INFILEP. If SHOW is set, prints file found if other than obvious.
   PREFERREDFILES is an association list of (file . filename) indicating an override of this default scheme; GETFILE will first check the indicated filename before trying anywhere else.
   To speed up repeated calls on the same file, GETFILE keeps track of the last several files it looked up; it will check this list (GETFILELST) before blindly searching other directories.

```
bvm: 30-May-78 22:44 [UTILITY]                                                        INFILEDIR
                                                                                    -------------

(INFILEDIR
  [LAMBDA (DIR NAME EXT)                                         (* bvm: "30-May-78 22:44")
    (PROG (JFN)
          (RETURN (COND
                    ((SETQ JFN (LGTJFN DIR NAME EXT))
                      (PROG1 [MKATOM (JFNS JFN NIL (CONSTANT (CONCAT]
                             (RLJFN JFN])
```

Called by: GETFILE

Explanation: Returns full name of file on directory DIR, where file is NAME.EXT (or NAME if EXT is nil, or NAME itself contains a "."). If DIR is NIL, connected directory is used. DIR may or may not begin with a "<".

```
bvm:  1-Jun-78 01:03 [UTILITY]                                                 OPENHASHFILEVARS
                                                                              ---------------------

(OPENHASHFILEVARS
  [LAMBDA (VARS WRITE? SAVE NOERROR)                             (* bvm: " 1-Jun-78 01:03")
    (for VAR inlist VARS bind FILE X collect (OR (COND
                                                   ([AND (SETQ FILE (GETATOMVAL VAR))
                                                         (NEQ FILE 'NOBIND)
                                                         (COND
                                                           (WRITE? (HASHFILEP (COND
                                                                                ((LISTP FILE)
                                                                                  (CAR FILE))
                                                                                (T FILE))
                                                                              'WRITE))
                                                           (T (OR (ARRAYP FILE)
                                                                  (LISTP FILE]
                                                                                (* It's already open)
                                                    FILE))
                                                 [SETATOMVAL VAR (COND
                                                                   ([NLISTP (SETQ FILE (CDR (FASSOC VAR MYCINHASHFILES]
                                                                     (OPENMYCINHASHFILE (OR FILE VAR)
                                                                                        WRITE? SAVE NOERROR VAR))
                                                                   ((SETQ X (OPENMYCINHASHFILE (CAR FILE)
                                                                                               WRITE? SAVE NOERROR VAR))
                                                                                (* Multiple file: open the first one.
                                                                                UGETHASHFILE will open the rest if
                                                                                needed)
                                                                     (CONS X (APPEND (CDR FILE]
                                                 (RETURN])
```

Calls: OPENMYCINHASHFILE

Called by: UGETHASHFILE

Globalvars: MYCINHASHFILES

Explanation: Opens the hashfiles indicated by VARS, a list of handles (or atom). Opens for write if WRITE? is set. If SAVE is set, adds entry to surrounding resetlst to restore the current state of the hashfiles (closed, open read...). Sets toplevel value of each of VARS to the corresponding hashfile datum, and returns a list of these data.
   The hashfile names are found in the association list MYCINHASHFILES. Any var not found there is treated as a filename itself. The "name" may be a list of names, in which case the value of the hashfile variable is a list of hashfiles, the first of which is opened (use UGETHASHFILE for these multiple guys; the other files are opened only as needed).
   NOERROR controls the situation when file can't be opened. If NIL, a ↑E is generated; if T, just quietly returns NIL; if a string, the string is printed before returning NIL.

```
(OPENMYCINHASHFILE
  [LAMBDA (FILE WRITE? SAVE NOERROR VAR)                                  (* bvm: " 1-Jun-78 00:59")
    (PROG (HASHFILE HELPFLAG)
          (RETURN (COND
                    ([AND (NEQ FILE T)
                          (SETQ FILE (GETFILE FILE T))
                          (OR (SETQ HASHFILE (HASHFILEP FILE WRITE?))
                              (AND [OR (EQ HASHCONFIRMFLG 'QUIET)
                                       (NOT WRITE?)
                                       (PROGN (TTYOUT1 "[Writing " FILE '%])
                                              (COND
                                                (HASHCONFIRMFLG (TERPRI T)
                                                                T)
                                                ((ASKYESNO NIL 'CONFIRM))
                                                (T (HELP]
                                   (NLSETQ (PROG ((BUSYCNT 0))
                                                 RETRY
                                                 (COND
                                                   ((NLSETQ (COND
                                                              ((AND WRITE? (SETQ HASHFILE (HASHFILEP FILE)))
                                                                                 (* File open for READ now, so close it
                                                                                 and reopen for write)
                                                               [AND SAVE (RESETSAVE NIL (LIST 'CLOSEHASHFILE HASHFILE
                                                                                              'READ]
                                                               (CLOSEHASHFILE FILE 'WRITE))
                                                              (T                 (* not open at all)
                                                                (SETQ HASHFILE (OPENHASHFILE FILE WRITE?))
                                                                (AND SAVE (RESETSAVE NIL
                                                                                     (LIST 'RESTOREHASHFILE VAR
                                                                                           HASHFILE)))
                                                                                 (* Give RESTOREHASHFILE the MULVAR to
                                                                                 clear, since file won't be reopened)
                                                                HASHFILE)))
                                                     (COND
                                                       ((NOT (ZEROP BUSYCNT))
                                                         (TTYOUT "free]")))
                                                     (RETURN))
                                                   ((EQ BUSYCNT 10)
                                                     (TTYOUT "timed out]")
                                                     (ERROR!)))
                                                 (COND
                                                   ((NEQ BUSYCNT 0)              (* we are waiting)
                                                     (PRIN1 '- T))
                                                   ((EQ (CAR (ERRORN))
                                                        9)                       (* File is busy; try waiting a bit)
                                                     (TTYOUT1 '%[ FILE " busy--"))
                                                   (T (ERROR!)))
                                                 (ADD1VAR BUSYCNT)
                                                 (DISMISS 1500)
                                                 (GO RETRY]
                      HASHFILE)
                    (T [COND
                         ((EQ FILE T)
                           (SETQ FILE VAR))
                         ((FMEMB (CAR (ERRORN))
                                 '(9 15 22 23))
                           (TTYOUT (ERSTR]
                       (COND
                         ((NULL NOERROR)                                 (* no provision for error, so report
                                                                         condition and abort)
                           (ERROR "Can't open file" FILE T))
                         ((NEQ NOERROR T)                                (* NOERROR = T means keep quiet;
                                                                         other values are error messages to
                                                                         print, before returning NIL)
                           (TTYOUT NOERROR])
----------
```

Calls: ASKYESNO GETFILE

Called by: OPENHASHFILEVARS UGETHASHFILE

Globalvars: HASHCONFIRMFLG

Explanation: Opens a single hashfile named FILE (searches for it with GETFILE), where WRITE?, SAVE, and NOERROR are as in OPENHASHFILEVARS. VAR is an atom, the hashfile variable which will be set to the hashfile datum (or data) and which here is used only when SAVE is set, to construct an appropriate reset expression (the var is reset to NIL when the hashfile is closed).
When opening for write, a warning will be printed, unless HASHCONFIRMFLG = QUIET. If HASHCONFIRMFLG is NIL, confirmation will be required. HASHCONFIRMFLG is initially T.
If the file is busy, will wait a while before giving up. A ↑E typed during this wait will abort it, resulting in the usual "file won't open" error condition.

```
bvm: 19-NOV-77 15:05 [UTILITY]                                                    PRINTPROP&VAL
                                                                                -----------------
(PRINTPROP&VAL
  [LAMBDA (PROP VALUE PROSEFLG)                                              (* bvm: "19-NOV-77 15:05")
    (PROG (**COMMENT**FLG TB)                                                (* Rebind **COMMENT**FLG so that TRANS's
                                                                              beginning with *'s show up)
          (SETQ TB (IPLUS (NCHARS PROP)
                          5))
          (TAB 2)
          (WRITE1 PROP ':)
          (COND
            (PROSEFLG (SPRINT VALUE 2 TB (IPLUS TB 3)))
            ((NLISTP VALUE)
              (SPRINT VALUE 2 TB NIL NIL NIL T))
            ([COND
                [(AND (CDR VALUE)
                      (FMEMB PROP RULEPTRS))                                 (* Display list of rules more concisely)
                  (SETQ VALUE (CONS 'Rules (RULENUMBERS VALUE]
                (T (AND (IGREATERP (NCHARS (CAR VALUE))
                                   7)
                        (NOTANY VALUE (FUNCTION LISTP]                       (* PRINTDEF might mess this up)
              (SPRINT VALUE 2 (IPLUS TB 4)
                      NIL NIL NIL T))
            (T (PRINTDEF VALUE TB)))
          (TERPRI])
----------
```

Called by: PRINTRECORD

Globalvars: RULEPTRS

Explanation: Prints property PROP and its VALUE in a nice property-style format. PROSEFLG is set if VALUE is the output of PROSE (as in rule translation).

```
(PRINTRECORD
  [LAMBDA (INSTANCE RECORDNAME)                                           (* bvm: "19-NOV-77 15:06")
    (PROG ((DEC (OR (RECLOOK RECORDNAME)
                    (ERROR RECORDNAME "not a record" T)))
           VALUE)
          (for FIELD in [DREVERSE (for F in (RECORDFIELDNAMES RECORDNAME) collect F when (ANYMEMB F (CADDR DEC]
             when (SETQ VALUE (RECORDACCESS FIELD INSTANCE DEC)) do (PRINTPROP&VAL FIELD VALUE])
----------
Calls:     PRINTPROP&VAL
```

Explanation: Prints the fields of a record. INSTANCE is the pointer to an instance of a record of type RECORDNAME.

```
(SPRINT
  [LAMBDA (LST INDENT PMAR LMAR LEVEL SEPR INDICATE)                      (* bvm: "16-NOV-77 08:28")
    (PROG  ((LEN (LINELENGTH))
            ENDWITH LSTWORD N PAREN SEPRFLG SEPRLEN)
           [OR (ARRAYP (GETATOMVAL 'SPRINTBITTABLE))
               (SETQ SPRINTBITTABLE (MAKEBITTABLE '(32 45 31]             (* (space, -, eol), for finding
                                                                          separator chars in strings)
           (SETQ SEPRLEN (SELECTQ SEPR
                                    (T                                    (* Means ", ")
                                       2)
                                    (NIL                                  (* default of space)
                                         1)
                                    (NCHARS SEPR)))
           [COND
             ((NOT INDENT)
               (SETQ INDENT 0))
             ((ZEROP INDENT))
             [(EQ INDENT T)
               (TERPRI)                                                   (* means start new line, at paragraph
                                                                          indentation)
               (COND
                 (PMAR (TAB PMAR))
                 (T (SETQ PMAR 0]
             ((MINUSP INDENT)                                             (* means begin on new line, unless
                                                                          already there)
               (TAB (OR PMAR (SETQ PMAR 0))
                     0))
             ((NOT (IGREATERP (SETQ N (IPLUS INDENT (POSITION)))
                              LEN))
               (TAB N))
             (T                                                           (* Too far over to space any, so start
                                                                          new line at appropriate indentation)
                (TAB (OR LMAR PMAR 0]
           (OR PMAR (SETQ PMAR INDENT))
           (OR LMAR (SETQ LMAR PMAR))
           (OR LEVEL (SETQ LEVEL 100))
           [COND
             [(NLISTP LST)                                                (* treat non-list as one-element list)
               (RETURN (SPRINT1 (FRPLACA (CONSTANT (CONS))
                                            LST]
             ((EQ (CAR LST)
                  '$I)                                                    (* ignore initial indents)
               (SETQ LST (CDR LST]
           [COND
             (INDICATE                                                    (* Show this is a list)
                         (SETQ PAREN '%()
                         (SETQ ENDWITH '%)]
           (SPRINT1 LST LEVEL)
           (COND
             (PAREN (PRIN1 PAREN)))
           (COND
             (ENDWITH (PRIN1 ENDWITH])
----------
Calls:     SPRINT1

Called by: ASKYESNO PRINTPROP&VAL

Freevars:  SPRINTBITTABLE
```

Explanation: Prints LST, initially spacing INDENT spaces and indenting by PMAR spaces. Linefeeds forced by line length use LMAR instead. LMAR defaults to PMAR defaults to INDENT defaults to zero. INDENT=T means start a new line at the paragraph indentation; a negative INDENT means start a new line if not there already. The special atom $L is used to represent carriage-return, linefeed. The EOL character (an atom) may also serve this function. $I causes linefeed plus indentation; $O (outdent) undoes a $I. If LST is not a list, it is treated as a one-element list. LST may contain strings, in which case they are broken at spaces as needed.

LEVEL is a printlevel parameter - lists at depth greater than LEVEL are printed as & (default is 100).

SEPR is a string or atom to prinl between elements of LST. Default is blank. T means comma, which will not be printed after the words 'and' and 'or'.

INDICATE is used to make SPRINT look like PRINT: if LST is a list, outer parens will appear, and strings in LST will be enclosed in quotes.

```
(SPRINT1
  [LAMBDA (LST LEVEL)                                                        (* bvm: "16-NOV-77 08:27")
    (PROG (WORD OPENQUOTE)
      TOP (SETQ WORD (CAR LST))
          (SETQ LST (CDR LST))
      SEL [COND
            ((STRINGP WORD)                                                  (* Print string, splitting as necessary)
              (SPRINTSTRING WORD INDICATE))
            ((LISTP WORD)                                                    (* Do lists recursively)
              [COND
                ((IGREATERP LEVEL 1)
                  (SPRINTPUNC '%( T)
                  (SPRINT1 WORD (SUB1 LEVEL))
                  (SPRINTPUNC '%)))
                (T (SPRINTATOM '&]
              (SETQ SEPRFLG T))
            (T (SELECTQ WORD
                        [($L %
)                                                                            (* End of line, possibly bare EOL)
                          (COND
                            ((NEQ (CAR LST)
                                 '$0)                                        (* End of line indicator can be ignored
                                                                             if followed by an outdent)
                              (SPRINTSEPR PMAR)
                              (SETQ SEPRFLG NIL]
                        ($I                                                  (* new paragraph, indented further)
                            (SPRINTSEPR (SETQ PMAR (IPLUS PMAR 3))
                                       T)
                            (SETQ LMAR (IPLUS LMAR 3))
                            (SETQ SEPRFLG NIL))
                        [$0                                                  (* outdent; ignore if final)
                            (COND
                              (LST (SPRINTSEPR (SETQ PMAR (IDIFFERENCE PMAR 3)))
                                   (SETQ LMAR (IDIFFERENCE LMAR 3))
                                   (SETQ SEPRFLG NIL]
                        ((%. , : ; %) %] ! ? 'S 's s ... ¦)                 (* "closing" punctuation)
                          (SPRINTPUNC WORD))
                        ((%( %[ ¬ ¦)                                         (* "Opening" punctuation)
                          (SPRINTPUNC WORD T))
                        [%"                                                  (* Figure out matching quotes)
                            (SPRINTPUNC WORD (SETQ OPENQUOTE (NOT OPENQUOTE]
                        (SPRINTATOM WORD]
          (SETQ LSTWORD WORD)
          (COND
            ((NOT LST)
              (RETURN))
            ((NLISTP LST)                                                    (* We just printed car of a dotted pair)
              (SPRINTATOM '%.)
              (SETQ WORD LST)
              (SETQ LST NIL)
              (GO SEL))
            (T (GO TOP])
----------
```

Calls: SPRINT1 SPRINTATOM SPRINTPUNC SPRINTSEPR SPRINTSTRING

Called by: SPRINT SPRINT1

Freevars: INDICATE LMAR LSTWORD PMAR SEPRFLG

Explanation: Recursive subfn of SPRINT which prints LST (recurring for any elements which are themselves lists and not in excess of LEVEL arg). Dispatches to other subfns according to each element of LST.

```
-------------------- [UTILITY]                                                                          SPRINTATOM
                                                                                                      --------------

(SPRINTATOM
  [LAMBDA (ATM)
    (PROG (POS (LIMIT (SPRINTCOUNT)))
          (COND
            ((IGREATERP (NCHARS ATM)
                        LIMIT)
              [COND
                ((AND (SETQ POS (STRPOSL SPRINTBITTABLE ATM))
                      (NOT (IGREATERP POS LIMIT)))                     (* Can be split up; let string handler
                                                                       do it)
                  (RETURN (SPRINTSTRING ATM]                           (* Just too big; start new line)
              (SPRINTSEPR LMAR))
            (T (SPRINTSEPR)))
          (PRIN1 ATM)
          (SETQ SEPRFLG T])
----------
Calls:     SPRINTCOUNT SPRINTSEPR SPRINTSTRING

Called by: SPRINT1

Freevars:  LMAR SEPRFLG SPRINTBITTABLE

Explanation:  Subfn of SPRINT to print, with appropriate separation and checks for fit, the single atom ATM.


bvm: 16-NOV-77 08:07 [UTILITY]                                                                         SPRINTCOUNT
                                                                                                      ---------------

(SPRINTCOUNT
  [LAMBDA NIL                                                          (* bvm: "16-NOV-77 08:07")
    (IDIFFERENCE LEN (IPLUS (POSITION)
                            (SELECTQ SEPRFLG
                                     (NIL 0)
                                     (%                                (* 2 spaces printed after period)
                                        2)
                                     SEPRLEN)
                            (COND
                              (PAREN                                   (* a backed up paren needs extra space)
                                     1)
                              (T 0))
                           2])
----------
Called by: SPRINTATOM SPRINTSTRING

Freevars:  LEN PAREN SEPRFLG SEPRLEN

Explanation:  Returns number of useable character positions on line, taking into account any saved chars/separators
           that we are already committed to printing.
```

```
(SPRINTPUNC
  [LAMBDA (CHAR OPEN?)                                               (* bvm: "16-NOV-77 08:26")
    (COND
      (OPEN?                                                         (* Save open paren for printing right
                                                                     before next word; we can check then if
                                                                     it will fit)
              (COND
                (PAREN                                               (* Old paren to clean up first)
                       (SPRINTSEPR)))
              (SETQ PAREN CHAR))
      (T                                                             (* No spacing before these)
        (COND
          ((IGREATERP (IPLUS (POSITION)
                             (SELECTQ CHAR
                                      (('S 's)                       (* do NCHARS in line)
                                        2)
                                      (... 3)
                                      1)
                             (COND
                               (PAREN 1)
                               (T 0)))
                      LEN)
            (TAB LMAR)))
        (COND
          (PAREN                                                     (* No separator printed, but we'd better
                                                                     clean up parens)
                 (PRIN1 PAREN)
                 (SETQ PAREN NIL)))
        (PRIN1 CHAR)
        (SETQ SEPRFLG (COND
            ((EQ CHAR '%.)
              '% )
            (T T])
----------
```

Calls: SPRINTSEPR

Called by: SPRINT1

Freevars: LEN LMAR PAREN SEPRFLG

Explanation: Handles punctuation for SPRINT. If OPEN? is true, treats CHAR as "opening" punctuation (spaces before, but not after, e.g. open paren); otherwise as "closing" (spaces after, not before). OPEN? is currently variable only for the character ".

```
(SPRINTSEPR
  [LAMBDA (NEWLINE DONTFORCE)                                   (* bvm: "19-NOV-77 15:08")
                                                                (* NEWLINE set if want new line after
                                                                separator)
    (SELECTQ SEPRFLG
             [%                                                 (* Last thing printed was a period, so
                                                                space twice)
                  (COND
                    ((NOT NEWLINE)
                      (SPACES 2]
             (NIL)
             (SELECTQ SEPR
                      [NIL                                      (* ordinary space; omit at end of line)
                           (OR NEWLINE (PRIN1 '% ]
                      [T                                        (* Comma, don't print after
                                                                conjunctions, don't space at EOL)
                         (COND
                           [(FMEMB LSTWORD '(and or))
                             (OR NEWLINE (PRIN1 '% ]
                           (NEWLINE (PRIN1 ',))
                           (T (PRIN1 ", "]
                      (PRIN1 SEPR)))
    (COND
      ((NOT NEWLINE)                                            (* just printed the sepr)
        )
      (DONTFORCE                                                (* this is mainly for indents;
                                                                don't go to newline if there's room
                                                                here)
                  (TAB NEWLINE))
      (T (TERPRI)
         (TAB NEWLINE 0)))
    (COND
      (PAREN                                                    (* print any backed-up paren)
             (PRIN1 PAREN)
             (SETQ PAREN NIL)))
    (SETQ SEPRFLG NIL])
----------
```

Called by: SPRINT1 SPRINTATOM SPRINTPUNC SPRINTSTRING

Freevars: LSTWORD PAREN SEPR SEPRFLG

Explanation: Subfn of SPRINT to print any separator chars needed (when SEPRFLG is set). Also includes any backed up PAREN. If NEWLINE is set, the separation is between lines, and NEWLINE is the tab stop for the new line.

```
(SPRINTSTRING
  [LAMBDA (STRING SHOWQUOTE)                                              (* bvm: " 9-JAN-78 22:58")
    [bind [#SPACES ←(IPLUS (SPRINTCOUNT)
                          (COND
                            (SHOWQUOTE -2)
                            (T 0]
          (#CHARS ←(NCHARS STRING))
          QFLG←SHOWQUOTE
          BRKPOS CH comment                                               (* Note that #SPACES and BRKPOS are
                                                                          always shorter than linelength, hence
                                                                          small integers. Thus the ADD1VARs work)

       while (COND
               ((ILESSP #CHARS #SPACES)
                 (STRPOS EOL STRING)))
       do (bind N←1 while [AND (SETQ N (STRPOSL SPRINTBITTABLE STRING (ADD1 N)))
                               (NOT (IGREATERP N #SPACES))
                               (COND
                                 ((EQ (SETQ CH (NTHCHAR STRING N))
                                      '-)
                                   (ILESSP (ADD1 N)
                                           #CHARS]
             do

          (* Set BRKPOS to be the last space before linelength runs out, or where EOL appears.
          #CHARS check assures that we don't break a short hyphenated atom over a line, e.g. CULTURE-1)


                (SETQ BRKPOS N)
             repeatwhile (NEQ CH EOL))
          (COND
            [BRKPOS (SPRINTSEPR)
                    (COND
                      (QFLG                                               (* Must indicate quotes)
                            (PRIN1 '%")
                            (SETQ QFLG NIL)))
                    (PRIN1 (SUBSTRING STRING 1 (COND
                                                 ((EQ CH '-)
                                                   BRKPOS)
                                                 (T (SUB1 BRKPOS)))
                                      (CONSTANT (CONCAT)               (* Scratch string, so we don't eat up
                                                                          too many string pointers)
                                                           ]
            (T (SETQ BRKPOS 0)))
          (repeatuntil (NEQ (SETQ CH (NTHCHAR STRING (ADD1VAR BRKPOS)))
                            '% ))                                         (* strip leading spaces from new piece)
          (COND
            ((EQ CH EOL)
              (ADD1VAR BRKPOS)))
          [COND
            ((EQ BRKPOS 1))
            ((NOT (IGREATERP (SETQ #CHARS (ADD1 (IDIFFERENCE #CHARS BRKPOS)))
                             0))
              (RETURN))
            (T (SETQ STRING (SUBSTRING STRING BRKPOS NIL (CONSTANT (CONCAT]
          (SPRINTSEPR LMAR)
          (SETQ #SPACES (IDIFFERENCE (IDIFFERENCE LEN LMAR)
                                     2))                                  (* This is new value of SPRINTCOUNT)
          (COND
            (SHOWQUOTE (SUB1VAR #SPACES)))
          (SETQ BRKPOS #SPACES)                                           (* Set LASTPOS to this in case there
                                                                          isn't a place to break)
       finally (COND
                 ((OR QFLG (NOT (ZEROP #CHARS)))
                   (SPRINTSEPR)
                   [COND
                     (QFLG (PRIN1 '%"]
                   (PRIN1 STRING]
    [COND
      (SHOWQUOTE                                                      (* Show closing quote)
                (PRIN1 '%"]
    (SETQ SEPRFLG (COND
        ((EQ (NTHCHAR STRING -1)
              '%.)                                                    (* want to space twice after this)
          '% )
        (T T])
----------
```

Calls: SPRINTCOUNT SPRINTSEPR

Called by: SPRINT1 SPRINTATOM

Globalvars: EOL

Freevars: LEN LMAR SEPRFLG SPRINTBITTABLE

Explanation: Subfn of SPRINT to print a string, splitting at spaces, hyphens and carriage returns as needed. SHOWQUOTE is true if the enclosing quotes are to be printed as well.

bvm: 18-JAN-78 23:38 [UTILITY]   TTYOUT

```
(TTYOUT
  [LAMBDA N                                                          (* bvm: "18-JAN-78 23:38")
                                                                     (* WRITE to tty)
    (for I from 1 to N do (WRITEARG (ARG N I)
                                    T))
    (TERPRI T])
----------
```

Calls: WRITEARG

Called by: CNET GETFILE OPENMYCINHASHFILE QSET!PARAMETERS RESIMULATE SET!PARAMETERS

Explanation: WRITE to terminal.

bvm: 1-Jun-78 01:04 [UTILITY]   UGETHASHFILE

```
(UGETHASHFILE
  [LAMBDA (HASHFILES KEY1 KEY2 ACCESS NOERROR)                                   (* bvm: " 1-Jun-78 01:04")
    (COND
      ([OR (NOT (LITATOM HASHFILES))
           (SETQ HASHFILES (OR (ARRAYP (GETATOMVAL HASHFILES))
                               (LISTP (GETATOMVAL HASHFILES))
                               (CAR (OPENHASHFILEVARS HASHFILES NIL NIL NOERROR]
                                                                                 (* HASHFILES can name a hashfile
                                                                                 variable)
         (SELECTQ ACCESS
                  (NIL                                                           (* normal get)
                       (SETQQ ACCESS RETRIEVE))
                  (LOOK                                                          (* See if there, but don't retrieve)
                        (SETQ ACCESS NIL))
                  NIL)
         (COND
           ((NLISTP HASHFILES)
             (LOOKUPHASHFILE KEY1 NIL HASHFILES ACCESS KEY2))
           (T (any (LOOKUPHASHFILE KEY1 NIL [OR (ARRAYP (CAR HASHFILES))
                                                 (CAR (FRPLACA HASHFILES (OR (OPENMYCINHASHFILE (CAR HASHFILES)
                                                                                                 NIL NIL NOERROR)
                                                                              (RETURN]
                                   ACCESS KEY2)
                   repeatwhile (SETQ HASHFILES (CDR HASHFILES]))
```

Calls: OPENHASHFILEVARS OPENMYCINHASHFILE

Called by: DISPLAYHELP

Explanation: Universal GETHASHFILE. Looks up in HASHFILES the entry indexed by KEY1 [and KEY2]. HASHFILES may be an open hashfile, list of hashfiles, or an atomic hashfile variable (i.e. anything that OPENHASHFILEVARS will accept). ACCESS is NIL for a normal GETHASHFILE; ACCESS=LOOK means lookup but don't return the value (just return T if ANY value found); other values of ACCESS are passed directly to LOOKUPHASHFILE. If HASHFILES is a hashfile variable (litatom), it will be opened. If HASHFILES is (or becomes thereby) a list, UGETHASHFILE looks up in each hashfile, returning the first non-NIL value found; if an element of this list is a filename instead of an open hashfile, it opens it and smashes the hashfile into the list. If HASHFILES is a non-list, behaves like a single hashfile lookup. NOERROR is passed to OPENHASHFILEVARS. KNOWNFILE that the function is on (used for advice in the editor). .If FN is a list, the above is done for each function in the list, and the result is the union of all the files. INTERNAL=T means KNOWNFILE is the full filename (otherwise needs to obtain the full name for comparison's sake). KNOWNFILE=T means simply print out ALL files containing FN(s).

bvm: 18-JAN-78 23:36 [UTILITY]   WRITE

```
(WRITE
  [LAMBDA N                                                                      (* bvm: "18-JAN-78 23:36")
                                                                                 (* WRITE PRIN1's its arguments to
                                                                                 primary output file, followed by EOL)
    (for I from 1 to N do (WRITEARG (ARG N I)))
    (TERPRI])
```

Calls: WRITEARG

Called by: ASKFORFILENAME ASKYESNO ATTRIBUTEP CILPARSE INTERPRET OBJECTP

Explanation: Takes arbitrary number of arguments, each of which is PRIN1ed to the primary output file, followed by crlf. If an argument is a list, it is MAPRINTed, i.e. the outer "parentheses" will not appear.

```
bvm: 18-JAN-78 23:37 [UTILITY]                                                        WRITE1
                                                                                    ----------
(WRITE1
  [LAMBDA N                                                  (* bvm: "18-JAN-78 23:37")
                                                             (* WRITE1 is WRITE without A TERPRI.)
    (for I from 1 to N do (WRITEARG (ARG N I])
----------
Calls:     WRITEARG

Called by: ASKFORFILENAME PRINTPROP&VAL

Explanation:  WRITE without the final crlf.
```

```
-------------------- [UTILITY]                                                         WRITE3
                                                                                    ----------
(WRITE3
  [LAMBDA N                                                  (* this is a WRITE1 that does PRIN3
                                                             instead of PRIN1, i.e. it ignores
                                                             linelength)
    (for I from 1 to N do (PRIN3 (ARG N I])
----------
Explanation:  A WRITE1 that ignores linelength, ie. does PRIN3's.
```

```
bvm: 22-JAN-78 23:27 [UTILITY]                                                       WRITEARG
                                                                                  ------------
(WRITEARG
  [LAMBDA (X FILE)                                           (* bvm: "22-JAN-78 23:27")
    (COND
      ((NLISTP X)
        (PRIN1 X FILE))
      (T (MAPRINT X FILE])
----------
Called by: DISPLAY TTYOUT WRITE WRITE1

Explanation:  If X is not a list, PRIN1's it to FILE, otherwise MAPRINT's it, so that the outer parens will not
           appear.
```

```
←(CNET)
                ----- CONTRACT NET Simulation -----


Reliability Mode  [ NO ] **
Nodes  [ 10 ] ** 5
Task time expansion factor  [ 100 ] **
Terminated contracts  [ 10 ] **
Default delay parameters  [ YES ] **
Display Parameters  [ YES ] **
Display statistics  [ YES ] **
Display banners  [ YES ] **
Display time  [ YES ] ** Yes
Display messages  [ NO ] ** Yes
Display internal events  [ NO ] ** Yes
Display display events  [ NO ] ** Yes
Display nodes  [ NO ] ** Yes
Diagnostic information to file  [ NO ] **
Initial Applications Function [$INITIALIZE] ** QINITIALIZE
Final Applications Function [$FINALIZE] ** QFINALIZE



  CONTRACT NET Simulation Parameters

  Normal Mode
  Number of Processor Nodes in Net: 5
  Applications time unit expansion: 100
  Contracts held in terminated state: 10

  CONTRACT NET Delay Parameters:
     Time to make a task announcement: 1
     Time before a task is reannounced: 1000
     Time to process a task announcement: 1
     Time to make a node availability announcement: 1
     Time to process a node availability announcement: 1
     Time to make a bid: 1
     Time to process a bid: 1
     Time to make a standard award: 1
     Time to process a standard award: 1
     Time to make a directed award: 1
     Time to process a directed award: 1
     Time to acknowledge a directed award: 1
     Time to process an acknowledgement: 1
     Time to make a report to another node: 1
     Time to process a report: 1
     Time to generate a termination: 1
     Time to process a termination: 1
     Time to generate a request: 1
     Time to process a request: 1
     Time to generate an information message: 1
     Time to process an information message: 1


::::::::::::::: Start of Simulation ::::::::::::::::


Number of Queens  [ 5 ] ** 4
Number of solutions  [ 1 ] ** 2
Search Strategy  [ 0 ] **
Report Strategy  [ 0 ] ** 1


: Time: 0


-- Node Status --
```

```
 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 1
 contract: 1
 internal event: contract processing

 From: 1

Started Processing Contract . 1


: Time: 200


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 1
 contract: 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 1


 node: 1
 contract: 1
 internal event: node update


: Time: 201


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (1 1)
 Suspended: NIL
 Terminated: NIL
 To: *
 From: 1
 Type: task announcement
 Contract: 1 1


: Time: 202


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (1 1)
 Suspended: NIL
 Terminated: NIL


 To: 1
 From: 2
 Type: bid
 Contract: 1 1


 To: 1
 From: 3
 Type: bid
 Contract: 1 1


 To: 1
 From: 4
 Type: bid
 Contract: 1 1


 To: 1
 From: 5
 Type: bid
 Contract: 1 1

: Time: 204


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 2
 From: 1
 Type: standard award
 Contract: 1 1
```

```
 node: 2
 contract: 1 1
 internal event: contract processing

 From: 2

Started Processing Contract 1 1


: Time: 400


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 1
 contract: 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 2


 node: 1
 contract: 1
 internal event: node update


: Time: 401


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (2 1)
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL
 To: *
 From: 1
 Type: task announcement
 Contract: 2 1


: Time: 402

-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (2 1)
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 1
 From: 3
 Type: bid
 Contract: 2 1


 To: 1
 From: 4
 Type: bid
 Contract: 2 1


 To: 1
 From: 5
 Type: bid
 Contract: 2 1


: Time: 404

-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 To: 3
 From: 1
 Type: standard award
 Contract: 2 1


 node: 3
 contract: 2 1
 internal event: contract processing

 From: 3

Started Processing Contract 2 1


: Time: 600

-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 1
 contract: 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 3


 node: 1
 contract: 1
 internal event: node update


: Time: 601


-- Node Status --
```

```
 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (3 1)
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: *
 From: 1
 Type: task announcement
 Contract: 3 1

: Time: 602

-- Node Status --

 Node 1
 Executing: (1)
 Ready: NIL
 Announced: (3 1)
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 1
 From: 4
 Type: bid
 Contract: 3 1

 To: 1
 From: 5
 Type: bid
 Contract: 3 1

: Time: 604

-- Node Status --

 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 2
 contract: 1 1
 internal event: node update


 To: 4
 From: 1
 Type: standard award
 Contract: 3 1

 From: 2

Generated Board--> Queen-rows: 1 3

 node: 2
 contract: 1 1
 internal event: node update


 node: 4
 contract: 3 1
 internal event: contract processing

 From: 4

Started Processing Contract 3 1

: Time: 605


-- Node Status --
```

```
 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: (1 1 1)
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: *
 From: 2
 Type: task announcement
 Contract: 1 1 1

: Time: 606

-- Node Status --

 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: (1 1 1)
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL
```

```
 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 2
 From: 5
 Type: bid
 Contract: 1 1 1

: Time: 608

-- Node Status --

 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 5
 From: 2
 Type: standard award
 Contract: 1 1 1
 node: 5
 contract: 1 1 1
 internal event: contract processing

 From: 5

Started Processing Contract 1 1 1


: Time: 800


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 1
 contract: 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 4


 node: 1
 contract: 1
 internal event: node update
```

```
 node: 1
 contract: 1
 internal event: node update

 From: 1

Suspended Contract 1


: Time: 801


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: *
 From: 1
 Type: task announcement
 Contract: 4 1
 To: 2
 From: 1
 Type: bid
 Contract: 1 1 1

: Time: 802

-- Node Status --

 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (4 1)
 Suspended: (1)
 Terminated: NIL

 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 To: 1
 From: 1
 Type: bid
 Contract: 4 1

: Time: 804

-- Node Status --

 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL

 node: 2
 contract: 1 1
 internal event: node update

 node: 4
 contract: 3 1
 internal event: node update

 To: 1
 From: 1
 Type: standard award
 Contract: 4 1

 From: 2

Generated Board--> Queen-rows: 1 4

 node: 2
 contract: 1 1
 internal event: node update

 From: 4

Generated Board--> Queen-rows: 3 1

 node: 4
 contract: 3 1
 internal event: node update
 node: 1
 contract: 4 1
 internal event: contract processing


 node: 2
 contract: 1 1
 internal event: node update

 From: 1

Started Processing Contract 4 1


: Time: 805


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: NIL
 Ready: NIL
 Announced: (2 1 1)
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: (1 3 1)
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: *
 From: 2
 Type: task announcement
 Contract: 2 1 1
```

```
 To: *
 From: 4
 Type: task announcement
 Contract: 1 3 1


 To: 2
 From: 2
 Type: bid
 Contract: 1 1 1


: Time: 806


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: NIL
 Ready: NIL
 Announced: (2 1 1)
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: (1 3 1)
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 2
 From: 2
 Type: bid
 Contract: 2 1 1
 To: 4
 From: 2
 Type: bid
 Contract: 1 3 1


: Time: 808


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 2
 From: 2
 Type: standard award
 Contract: 2 1 1


 To: 2
 From: 4
 Type: standard award
 Contract: 1 3 1


 node: 2
 contract: 2 1 1
 internal event: contract processing

 From: 2

Started Processing Contract 2 1 1

: Time: 904

-- Node Status --

 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 3
 contract: 2 1
 internal event: node update

 From: 3

Generated Board--> Queen-rows: 2 4


 node: 3
 contract: 2 1
 internal event: node update


 node: 3
 contract: 2 1
 internal event: node update

 From: 3

Suspended Contract 2 1

: Time: 905

-- Node Status --
```

```
Node 1
Executing: (4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL


Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL


Node 3
Executing: NIL
Ready: NIL
Announced: (1 2 1)
Suspended: (2 1)
Terminated: NIL


Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL


Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL


To: *
From: 3
Type: task announcement
Contract: 1 2 1


To: 2
From: 3
Type: bid
Contract: 1 1 1


; Time: 906

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL

Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL


Node 3
Executing: NIL
Ready: NIL
Announced: (1 2 1)
Suspended: (2 1)
Terminated: NIL


Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL


Node 5
Executing: (1 1 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL


To: 3
From: 3
Type: bid
Contract: 1 2 1

; Time: 908

-- Node Status --

Node 1
Executing: (4 1)
Ready: NIL
Announced: NIL
Suspended: (1)
Terminated: NIL


Node 2
Executing: (2 1 1)
Ready: (1 3 1)
Announced: NIL
Suspended: (1 1)
Terminated: NIL


Node 4
Executing: (3 1)
Ready: NIL
Announced: NIL
Suspended: NIL
Terminated: NIL
```

```
 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 To: 3
 From: 3
 Type: standard award
 Contract: 1 2 1


 node: 3
 contract: 1 2 1
 internal event: contract processing

 From: 3

Started Processing Contract 1 2 1


: Time: 1004


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL
```

```
 node: 1
 contract: 4 1
 internal event: node update

 From: 2

Suspended Contract 1 1

 From: 1

Generated Board--> Queen-rows: 4 1


 node: 1
 contract: 4 1
 internal event: node update


: Time: 1005


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL
 To: *
 From: 1
 Type: task announcement
 Contract: 1 4 1


: Time: 1104


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 4
 contract: 3 1
 internal event: node update


: Time: 1108


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 5
 Executing: (1 1 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 node: 5
 contract: 1 1 1
 internal event: node update


 node: 2
 contract: 2 1 1
 internal event: node update


 node: 3
 contract: 1 2 1
 internal event: node update


 node: 5
 contract: 1 1 1
 internal event: node update

 From: 2

Generated Board--> Queen-rows: 1 4 2


 node: 2
 contract: 2 1 1
 internal event: node update

 From: 3

Generated Board--> Queen-rows: 2 4 1
```

```
 node: 3
 contract: 1 2 1
 internal event: node update

 From: 5

Terminated Contract 1 1 1


: Time: 1109


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 3 1)
 Announced: (1 2 1 1)
 Suspended: (1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: (1 1 2 1)
 Suspended: (2 1)
 Terminated: NIL


 To: 2
 From: 5
 Type: final report
 Contract: 1 1 1


 To: *
 From: 2
 Type: task announcement
 Contract: 1 2 1 1


 To: *
 From: 3
 Type: task announcement
 Contract: 1 1 2 1


: Time: 1110


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: (1 2 1 1)
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: (1 1 2 1)
 Suspended: (2 1)
 Terminated: NIL


 To: 2
 From: 4
 Type: bid
 Contract: 1 2 1 1


 To: 2
 From: 5
 Type: bid
 Contract: 1 2 1 1


 To: 3
 From: 4
 Type: bid
 Contract: 1 1 2 1


 To: 3
 From: 5
 Type: bid
 Contract: 1 1 2 1


: Time: 1112


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL
 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 To: 4
 From: 2
 Type: standard award
 Contract: 1 2 1 1


 To: 4
 From: 3
 Type: standard award
 Contract: 1 1 2 1


 node: 4
 contract: 1 2 1 1
 internal event: contract processing

 From: 4

Started Processing Contract 1 2 1 1


: Time: 1200


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL
```

```
 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 node: 1
 contract: 1 1
 internal event: bid check


: Time: 1204


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 node: 2
 contract: 1 1
 internal event: pseudo contract


 node: 3
 contract: 1 1
 internal event: pseudo contract


 node: 4
 contract: 1 1
 internal event: pseudo contract
 node: 5
 contract: 1 1
 internal event: pseudo contract


 node: 1
 contract: 4 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 4 2


 node: 1
 contract: 4 1
 internal event: node update


: Time: 1205


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (2 4 1) (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 To: *
 From: 1
 Type: task announcement
 Contract: 2 4 1


: Time: 1206


-- Node Status --

 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (2 4 1) (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 To: 1
 From: 5
 Type: bid
 Contract: 2 4 1


: Time: 1208


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL
 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 To: 5
 From: 1
 Type: standard award
 Contract: 2 4 1


 node: 5
 contract: 2 4 1
 internal event: contract processing

 From: 5

Started Processing Contract 2 4 1


: Time: 1304


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL

 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)

 From: 4

Suspended Contract 3 1


: Time: 1308


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: (1 1) (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 2
 contract: 2 1 1
 internal event: node update


: Time: 1309


-- Node Status --
```

```
 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: (2 1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 2
 contract: 1 1
 internal event: node update


 node: 2
 contract: 1 1
 internal event: node update

 From: 2

Suspended Contract 1 1


: Time: 1310


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL
```

```
 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 2
 contract: 1 3 1
 internal event: contract processing

 From: 2

Started Processing Contract 1 3 1


: Time: 1400


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL
 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 1
 contract: 2 1
 internal event: bid check


: Time: 1404

-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL
```

```
 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 3
 contract: 2 1
 internal event: pseudo contract


 node: 4
 contract: 2 1
 internal event: pseudo contract


 node: 5
 contract: 2 1
 internal event: pseudo contract


 node: 1
 contract: 4 1
 internal event: node update


: Time: 1408

-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL
 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 3
 contract: 1 2 1
 internal event: node update


: Time: 1600


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 1
 contract: 3 1
 internal event: bid check


: Time: 1604


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 4
 contract: 3 1
 internal event: pseudo contract


 node: 5
 contract: 3 1
 internal event: pseudo contract


 node: 2
 contract: 1 1 1
 internal event: bid check


: Time: 1608


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL
 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 5
 contract: 1 1 1
 internal event: pseudo contract


: Time: 1612


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 4
 Executing: (1 2 1 1)
 Ready: (1 1 2 1)
 Announced: NIL
 Suspended: (3 1)
 Terminated: NIL


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 4
 contract: 1 2 1 1
 internal event: node update
```

```
 node: 4
 contract: 1 2 1 1
 internal event: node update

 From: 4

Terminated Contract 1 2 1 1


: Time: 1613


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 1) (2 1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 To: 2
 From: 4
 Type: final report
 Contract: 1 2 1 1


 node: 4
 contract: 1 1 2 1
 internal event: contract processing

 From: 4

Started Processing Contract 1 1 2 1


: Time: 1708


-- Node Status --
```

```
 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 Node 5
 Executing: (2 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 1)


 node: 5
 contract: 2 4 1
 internal event: node update


 node: 5
 contract: 2 4 1
 internal event: node update

 From: 5

Terminated Contract 2 4 1


: Time: 1709


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL

 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 To: 1
 From: 5
 Type: final report
 Contract: 2 4 1


: Time: 1710


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 1
 contract: 4 1
 internal event: node update


 node: 1
 contract: 4 1
 internal event: node update

 From: 1

Suspended Contract 4 1


: Time: 1800


-- Node Status --
```

```
 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 1
 contract: 4 1
 internal event: bid check

: Time: 1803

-- Node Status --

 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)

 node: 1
 contract: 1 1 1
 internal event: pseudo contract

: Time: 1804

-- Node Status --

 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 1
 contract: 4 1
 internal event: pseudo contract


 node: 2
 contract: 2 1 1
 internal event: bid check


 node: 4
 contract: 1 3 1
 internal event: bid check

: Time: 1807

-- Node Status --

 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL
 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 2
 contract: 1 1 1
 internal event: pseudo contract


: Time: 1808


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 2
 contract: 2 1 1
 internal event: pseudo contract


 node: 2
 contract: 1 3 1
 internal event: pseudo contract

 From: 2

Suspended Contract 2 1 1


: Time: 1810


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 3 1)
 Ready: (2 1 1)
 Announced: NIL
 Suspended: (1 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 2
 contract: 1 3 1
 internal event: node update

 From: 2

Generated Board--> Queen-rows: 3 1 4


 node: 2
 contract: 1 3 1
 internal event: node update


 node: 2
 contract: 1 3 1
 internal event: node update

 From: 2

Suspended Contract 1 3 1


: Time: 1811


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (2 1 1)
 Ready: NIL
 Announced: (1 1 3 1)
 Suspended: (1 3 1) (1 1)
 Terminated: NIL
 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 To: *
 From: 2
 Type: task announcement
 Contract: 1 1 3 1


 node: 2
 contract: 2 1 1
 internal event: node update


 node: 2
 contract: 2 1 1
 internal event: node update


 node: 2
 contract: 2 1 1
 internal event: node update

 From: 2

Terminated Contract 2 1 1


: Time: 1812


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: NIL
 Ready: NIL
 Announced: (1 1 3 1)
 Suspended: (1 3 1) (1 1)
 Terminated: (2 1 1)


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 To: 2
 From: 1
 Type: bid
 Contract: 1 1 3 1


 To: 2
 From: 3
 Type: bid
 Contract: 1 1 3 1


 To: 2
 From: 5
 Type: bid
 Contract: 1 1 3 1


 To: 2
 From: 2
 Type: final report
 Contract: 2 1 1


: Time: 1813


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 2
 Executing: (1 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (2 1 1)


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 2
 contract: 1 1
 internal event: node update


 node: 2
 contract: 1 1
 internal event: node update
 node: 2
 contract: 1 1
 internal event: node update

 From: 2

Terminated Contract 1 1


: Time: 1814


-- Node Status --


 Node 1
 Executing: NIL
 Ready: NIL
 Announced: (1 4 1)
 Suspended: (4 1) (1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 To: 1
 From: 2
 Type: standard award
 Contract: 1 1 3 1


 To: 1
 From: 2
 Type: final report
 Contract: 1 1


 node: 1
 contract: 1 1 3 1
 internal event: contract processing

 From: 1

Started Processing Contract 1 1 3 1


: Time: 1904


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL
```

```
 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 3
 contract: 1 2 1
 internal event: bid check

 From: 1

Suspended Contract 4 1


: Time: 1907


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 3
 contract: 1 1 1
 internal event: pseudo contract


: Time: 1908


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)
 node: 3
 contract: 1 2 1
 internal event: pseudo contract

 From: 3

Suspended Contract 1 2 1


: Time: 2004


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 1
 contract: 1 4 1
 internal event: bid check


: Time: 2005


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)
```

```
 To: *
 From: 1
 Type: task announcement
 Contract: 1 4 1


: Time: 2006


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: (1 4 1)
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 To: 1
 From: 2
 Type: bid
 Contract: 1 4 1


 To: 1
 From: 3
 Type: bid
 Contract: 1 4 1


 To: 1
 From: 5
 Type: bid
 Contract: 1 4 1


: Time: 2008


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)
 To: 2
 From: 1
 Type: standard award
 Contract: 1 4 1


 node: 2
 contract: 1 4 1
 internal event: contract processing

 From: 2

Started Processing Contract 1 4 1


: Time: 2013


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 Node 4
 Executing: (1 1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (3 1)
 Terminated: (1 2 1 1)


 node: 4
 contract: 1 1 2 1
 internal event: node update

 From: 4

Generated Board--> Queen-rows: 2 4 1 3


 node: 4
 contract: 1 1 2 1
 internal event: node update
```

```
 node: 4
 contract: 1 1 2 1
 internal event: node update

 From: 4

Terminated Contract 1 1 2 1


: Time: 2014


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 To: 3
 From: 4
 Type: final report
 Contract: 1 1 2 1


: Time: 2015


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 Node 3
 Executing: (1 2 1)
 Ready: NIL
 Announced: NIL
 Suspended: (2 1)
 Terminated: NIL
 node: 3
 contract: 1 2 1
 internal event: node update


 node: 3
 contract: 1 2 1
 internal event: node update


 node: 3
 contract: 1 2 1
 internal event: node update

 From: 3

Terminated Contract 1 2 1


: Time: 2016


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 To: 3
 From: 3
 Type: final report
 Contract: 1 2 1


: Time: 2017


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)
```

```
 Node 3
 Executing: (2 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 2 1)


 node: 3
 contract: 2 1
 internal event: node update


 node: 3
 contract: 2 1
 internal event: node update


 node: 3
 contract: 2 1
 internal event: node update

 From: 3

Terminated Contract 2 1


: Time: 2018


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 To: 1
 From: 3
 Type: final report
 Contract: 2 1


: Time: 2108


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL
 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 node: 2
 contract: 1 2 1 1
 internal event: bid check


 node: 3
 contract: 1 1 2 1
 internal event: bid check


: Time: 2112


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 node: 4
 contract: 1 2 1 1
 internal event: pseudo contract


 node: 5
 contract: 1 2 1 1
 internal event: pseudo contract


 node: 4
 contract: 1 1 2 1
 internal event: pseudo contract


 node: 5
 contract: 1 1 2 1
 internal event: pseudo contract


: Time: 2114


-- Node Status --


 Node 1
 Executing: (1 1 3 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: NIL


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 node: 1
 contract: 1 1 3 1
 internal event: node update

 From: 1

Generated Board--> Queen-rows: 3 1 4 2


 node: 1
 contract: 1 1 3 1
 internal event: node update


 node: 1
 contract: 1 1 3 1
 internal event: node update

 From: 1

Terminated Contract 1 1 3 1


: Time: 2115


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: (4 1)
 Terminated: (1 1 3 1)


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 3 1)
 Terminated: (1 1) (2 1 1)


 To: 2
 From: 1
 Type: final report
 Contract: 1 1 3 1
 node: 1
 contract: 1
 internal event: node update


 node: 1
 contract: 1
 internal event: node update

 From: 1

Suspended Contract 1


: Time: 2204


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 node: 1
 contract: 2 4 1
 internal event: bid check


: Time: 2208


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 node: 5
 contract: 2 4 1
 internal event: pseudo contract


: Time: 2408


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1) (2 1 1)
```

```
 node: 2
 contract: 1 4 1
 internal event: node update

 From: 2

Generated Board--> Queen-rows: 4 1 3


 node: 2
 contract: 1 4 1
 internal event: node update


: Time: 2409


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: (1 1 4 1)
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 To: *
 From: 2
 Type: task announcement
 Contract: 1 1 4 1


: Time: 2410


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: (1 1 4 1)
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 To: 2
 From: 1
 Type: bid
 Contract: 1 1 4 1


 To: 2
 From: 3
 Type: bid
 Contract: 1 1 4 1


 To: 2
 From: 4
 Type: bid
 Contract: 1 1 4 1
 To: 2
 From: 5
 Type: bid
 Contract: 1 1 4 1


: Time: 2412


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 To: 1
 From: 2
 Type: standard award
 Contract: 1 1 4 1


 node: 1
 contract: 1 1 4 1
 internal event: contract processing

 From: 1

Started Processing Contract 1 1 4 1


: Time: 2508


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1) (4 1)
 Terminated: (1 1 3 1)


 Node 2
 Executing: (1 4 1)
 Ready: (1 3 1)
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1) (2 1 1)


 node: 2
 contract: 1 4 1
 internal event: node update


: Time: 2509


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1) (4 1)
 Terminated: (1 1 3 1)


 Node 2
 Executing: (1 3 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1 4 1)
 Terminated: (1 1) (2 1 1)


 node: 2
 contract: 1 3 1
 internal event: node update


 node: 2
 contract: 1 3 1
 internal event: node update


 node: 2
 contract: 1 3 1
 internal event: node update

 From: 2

Terminated Contract 1 3 1


: Time: 2510


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1) (4 1)
 Terminated: (1 1 3 1)


 To: 4
 From: 2
 Type: final report
 Contract: 1 3 1


: Time: 2511


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1) (4 1)
 Terminated: (1 1 3 1)
 Node 4
 Executing: (3 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 1 2 1) (1 2 1 1)


 node: 4
 contract: 3 1
 internal event: node update


 node: 4
 contract: 3 1
 internal event: node update


 node: 4
 contract: 3 1
 internal event: node update

 From: 4

Terminated Contract 3 1


: Time: 2512


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1) (4 1)
 Terminated: (1 1 3 1)


 To: 1
 From: 4
 Type: final report
 Contract: 3 1


: Time: 2810


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: (1 1 3 1)
```

```
 node: 2
 contract: 1 1 3 1
 internal event: bid check


: Time: 2814


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: (1 1 3 1)


 node: 1
 contract: 1 1 3 1
 internal event: pseudo contract


 node: 3
 contract: 1 1 3 1
 internal event: pseudo contract


 node: 5
 contract: 1 1 3 1
 internal event: pseudo contract


: Time: 2912


-- Node Status --


 Node 1
 Executing: (1 1 4 1)
 Ready: (1)
 Announced: NIL
 Suspended: (4 1)
 Terminated: (1 1 3 1)


 node: 1
 contract: 1 1 4 1
 internal event: node update


 node: 1
 contract: 1 1 4 1
 internal event: node update

 From: 1

Terminated Contract 1 1 4 1


: Time: 2913


-- Node Status --
```

```
 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: (4 1)
 Terminated: (1 1 4 1) (1 1 3 1)


 To: 2
 From: 1
 Type: final report
 Contract: 1 1 4 1


 node: 1
 contract: 1
 internal event: node update


 node: 1
 contract: 1
 internal event: node update

 From: 1

Suspended Contract 1


: Time: 2914


-- Node Status --


 Node 2
 Executing: (1 4 1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (1 3 1) (1 1) (2 1 1)


 node: 2
 contract: 1 4 1
 internal event: node update


 node: 2
 contract: 1 4 1
 internal event: node update


 node: 2
 contract: 1 4 1
 internal event: node update

 From: 2

Terminated Contract 1 4 1


: Time: 2915


-- Node Status --

 To: 1
 From: 2
 Type: final report
 Contract: 1 4 1


: Time: 2916


-- Node Status --


 Node 1
 Executing: (4 1)
 Ready: NIL
 Announced: NIL
 Suspended: (1)
 Terminated: (1 1 4 1) (1 1 3 1)


 node: 1
 contract: 4 1
 internal event: node update


 node: 1
 contract: 4 1
 internal event: node update


 node: 1
 contract: 4 1
 internal event: node update

 From: 1

Terminated Contract 4 1


: Time: 2917


-- Node Status --


 To: 1
 From: 1
 Type: final report
 Contract: 4 1


: Time: 2918


-- Node Status --


 Node 1
 Executing: (1)
 Ready: NIL
 Announced: NIL
 Suspended: NIL
 Terminated: (4 1) (1 1 4 1) (1 1 3 1)
 node: 1
 contract: 1
 internal event: node update


 node: 1
 contract: 1
 internal event: node update


 node: 1
 contract: 1
 internal event: node update

 From: 1

Terminated Contract 1


: Time: 2919


-- Node Status --


 To: 0
 From: 1
 Type: final report
 Contract: 1

Solutions Found:
 Queen-rows: 3 1 4 2
 Queen-rows: 2 4 1 3


: Time: 3004


-- Node Status --


 node: 1
 contract: 1 4 1
 internal event: bid check


: Time: 3008


-- Node Status --


 node: 2
 contract: 1 4 1
 internal event: pseudo contract


 node: 3
 contract: 1 4 1
 internal event: pseudo contract
```

```
 node: 5
 contract: 1 4 1
 internal event: pseudo contract

 From: 2

Suspended Contract 1 4 1


: Time: 3408


-- Node Status --


 node: 2
 contract: 1 1 4 1
 internal event: bid check


: Time: 3412


-- Node Status --


 node: 1
 contract: 1 1 4 1
 internal event: pseudo contract


 node: 3
 contract: 1 1 4 1
 internal event: pseudo contract


 node: 4
 contract: 1 1 4 1
 internal event: pseudo contract


 node: 5
 contract: 1 1 4 1
 internal event: pseudo contract



::::::::::::::::: End of Simulation :::::::::::::::::


  Time Units to Completion:2919

  Processor Node Utilization Statistics
  -------------------------------------

     Node        Utilization
       1          .7553957
       2          .7211374
       3          .3429257
       4          .4799589
       5          .3422405

  Mean Processor Node Utilization: .5283316
  Standard Deviation: .2000483

Another task  [ YES ] ** No
NIL
```