12-1987

Ludus Latrunculorum

Heather Anne Laudan

Heather Laudan
Senior Thesis
(program)
Fall '87

HEATHER LAUDAN:
Ludus Latrunculorum

Fall 1987

```
program thesis;
{$i graph.p}

label quit ;

const allstars = '**********************************************************************
******************' ;

type
  mwtype = (man,woman);

  piecetype = record
    morw : mwtype;
    role : (main,chorus);
    cancall : boolean ;
   end ;

  square = record
    full : boolean ;
    piece : piecetype ;
  end;

  letterset = set of 'A'..'z';
  godtype = (ares,aphrodite,notthere) ;
  boardtype = array [1..8,1..8] of square ;
  type callarray = array [1..2] of integer ;

var
  board : boardtype ;
  side, action, newgame, newboard : char ;
  callcount : callarray ;
  lastnum, oldrow, oldcol, newrow, newcol, numplayers : integer ;
  godcalled : godtype ;
  noneleft, change : boolean ;
  oldsquare, newsquare : square ;
  outloop1, outloop2, moveset : letterset ;


function keyin : char ;

var c:char ;

begin
  repeat until keypressed ;
  read (kbd, c) ;
  keyin := c ;
end ;
```

```
procedure play (note, time : integer) ;

begin
  sound (note) ;
  delay (time) ;
  nosound ; ;
  delay (15) ;
end ;

procedure rest ;

begin
  delay (200) ;
end ;

procedure music ;

const
  n8 = 400 ;
  e1 = 125 ;
  fs = 140 ;
  g = 147 ;
  a = 159 ;
  b = 170 ;
  cs = 180 ;
  d = 185 ;
  e2 = 193 ;

var n4, n3 : integer ;
  more : char ;

begin
  textmode(c80) ;
  n4 := 2 * n8 ;
  n3 := 3 * n8 ;
  textbackground (black) ;
  textcolor (magenta) ;
  play (a, n8) ;
  play (e2, n4) ;
  play (e2, n3) ;
  play (cs, n8) ;
  play (d, n8) ;
  play (e2, n8) ;
  play (d, n3) ;
  rest ;
  play (cs, n4) ;
  play (d, n8) ;
  play (e2, n8) ;
  play (a, n8) ;
  play (cs, n8) ;
  play (b, n8) ;
  play (a, n4) ;
  play (b, n8) ;
  play (g, n4) ;
  rest ;
  play (a, n8) ;
  play (cs, n8) ;
  play (e2, n8) ;
  play (d, n8) ;
  play (cs, n8) ;
  play (d, n8) ;
  play (cs, n8) ;
  play (a, n4) ;
  play (b, n8) ;
  play (g, n4) ;
```

```
  rest ;
  play (a, n8) ;
  play (cs, n8) ;
  play (b, n8) ;
  play (d, n8) ;
  play (e2, n8) ;
  play (cs, n8) ;
  play (a, n8) ;
  play (a, n4) ;
  play (a, n8) ;
  play (fs, n8) ;
  play (e1, n3) ;
  rest ;
end ;

procedure smallsound ;

const
  n8 = 400 ;
  e1 = 125 ;
  fs = 140 ;
  g = 147 ;
  a = 159 ;
  b = 170 ;
  cs = 180 ;
  d = 185 ;
  e2 = 193 ;

var n4, n3 : integer ;
  more : char ;

begin
  n4 := 2 * n8 ;
  n3 := 3 * n8 ;
  play (a, n8) ;
  play (cs, n8) ;
  play (b, n8) ;
  play (d, n8) ;
  play (e2, n8) ;
  play (cs, n8) ;
  play (a, n8) ;
  play (a, n4) ;
  play (a, n8) ;
  play (fs, n8) ;
  play (e1, n3) ;
  rest ;
end ;

procedure writestart ;

var cont : char ;
    skipset : letterset ;

begin
  textmode (c80) ;
  textbackground (black) ;
  textcolor (magenta) ;
  skipset := ['s','S'];
  clrscr ;
  writeln ;
  for lastnum := 1 to 10 do
    write (allstars) ;
  writeln ('                              Ludus Latrunculorum');
  writeln ('                               by Heather Laudan') ;
  for lastnum := 1 to 10 do
    write (allstars) ;
```

```
  music ;
  clrscr ;
  writeln (' If you would like to skip the introduction enter the letter s.');
  writeln (' To continue - press any other key.') ;
  cont := keyin ;
  if cont in skipset
    then exit ;
  writeln ;
  writeln (' This is a game loosely based on an ancient game played by');
  writeln (' Greeks and Romans.  The Romans called the game Ludus ');
  writeln (' Latrunculorum, or robbers.  Remnants of game pieces found') ;
  writeln (' in tombs and ruins of ancient buildings, pictures depicting');
  writeln (' play, and references in poetry have given us the few');
  writeln (' details which we have about the game.  The board is a square');
  writeln (' grid of square blocks upon which two opposing teams match forces');
  writeln (' Pieces are captured and removed from play when they are');
  writeln (' surrounded on two opposite sides by two pieces from the');
  writeln (' other side.  It is debated whether there was just one type of ');
  writeln (' move for all pieces, or possibly two types of pieces on each side,
');
  writeln (' with different allowable moves for the different pieces.');
  writeln (' This game follows the latter.  The remaining details of the');
  writeln (' ancient game are as yet undiscovered, but have been liberally');
  writeln (' interpreted with the help of the Greek comic playwrite,');
  writeln (' Aristophanes, and his play entitled Lysistrata.  The Lysistrata is'
);
  writeln (' the basis for this strategic battle, with the opposing sides not');
  writeln (' soldiers of opposing city - states fighting for territory or');
  writeln ;
  writeln (' (press any key to see more.)') ;
  cont := keyin ;
  clrscr ;
  writeln ;
  writeln (' glory, but opposing sexes: men versus women in a battle');
  writeln (' for international peace as well as domestic tranquility.  The ');
  writeln (' women of warring city-states, tired of the tolls the war is');
  writeln (' taking on their country and their private lives, unite against');
  writeln (' all their husband-soldiers and stage a sex strike until peace');
  writeln (' is declared.') ;
  writeln ;
  writeln (' (press any key to see more.)') ;
  cont := keyin ;
  clrscr ;
  writeln ;
  writeln ('  In this game the game board or the screen represents the ');
  writeln ('  scene of most of the action of the Lysistrata. the Parthenon.');
  writeln (' The opposing sides are the women led by the militant matron Lysistr
ata, ') ;
  writeln (' vs. the men.  There are two types of pieces for each side,');
  writeln (' 12 of each type.  The shaded pieces have the most freedom and');
  writeln (' power in their movement as they are allowed to move in an') ;
  writeln (' L- shape in any direction.  These pieces represent the the');
  writeln (' main characters of the comedy, the young soldiers and their');
  writeln (' wives.  The outlined pieces represent the constantly bickering');
  writeln (' old men and women of the chorus.  The chorus pieces are allowed');
  writeln (' to move only one square at a time, either forward, backward.');
  writeln (' left, or right. ') ;
  writeln ;
  writeln (' (press any key to see more.)') ;
  cont := keyin ;
  clrscr ;
  writeln ;
  writeln (' The object of the game is to eliminate all pieces of the opposing')
;
  writeln (' team.  Each side is helped toward its object by a patron god.');
  writeln (' The women are backed by Aphrodite, goddess of love.  The soldiers'
```

```
');
  writeln (' patron god is Ares, god of war.  Occasionally during play of');
  writeln (' the game a side may choose to call upon the gods to interfere');
  writeln (' on their behalf.  This may or may not act in that sides'' favor,');
  writeln (' or the gods may not deign to help anyone.') ;
  writeln (' Whenever a god-call occurs Ares and Aphrodite (who are fairly');
  writeln (' evenly matched for this game) battle it out in the heavens.');
  writeln (' When a victor is determined, the side which is backed by the ');
  writeln (' victor is allowed to remove any one piece from the other side');
  writeln (' and play continues.  Each side starts with 3 god calls.  Whenever')
;
  writeln (' a chorus member first reaches the home side of its opposition');
  writeln (' that team acquires another god-call.  Each chorus member can add on
ly');
  writeln (' one god-call to its side. ') ;
  writeln ;
  writeln (' (press any key to see more.)') ;
  cont := keyin ;
  clrscr ;
  writeln ;
  writeln(' Pieces are removed when they are surrounded on two opposite');
  writeln(' sides by 2 pieces from the opposing side.  A piece in a corner is');
  writeln(' surrounded if three opposing pieces are all around it.  Teams');
  writeln(' alternate turns moving one piece in each turn.  ');
  writeln(' More than one piece can be eliminated at once');
  writeln(' by surrounding a row (or column) of contiguous pieces at');
  writeln(' its two ends.');
  writeln ;
  writeln (' Play may be conducted between two players, or one player');
  writeln (' against the computer.(not yet)  The players choose which sides they
');
  writeln (' want, and an initial god-call determines who goes first.');
  writeln ;
  writeln (' (press any key to start the game.)') ;
  cont := keyin ;
  clrscr ;
end ;

procedure oneplayer  (var  side : char ) ;

begin
  writeln ('What side will you take against me?') ;
  writeln ('Type m - men or w - women') ;
  side := keyin ;
  if (side = 'm') or (side = 'M')
    then
      begin
        writeln (' All right - your patron is Ares.') ;
        writeln ( 'Remember love conquers all!')
      end
    else writeln('Your patron is Aphrodite.');
  writeln ;
end ;

procedure twoplayers (var side1 : char ) ;

begin
  writeln (' Player 1, which side will you take?') ;
  writeln (' Enter the letter m or w.') ;
  side1 := keyin ;
  if (side1 = 'm') or (side1 = 'M')
    then
      begin
        writeln (' Player 1, your patron is Ares .') ;
        writeln (' Player 2, Aphrodite loves you .')
      end
```

```
    else
      begin
        writeln (' Player 1, your patron is Aphrodite .') ;
        writeln (' Player 2, Ares is at your back raging to do war.')
      end ;
  delay (2500)
end ;

procedure getnum ( var  side1 : char  ;
                   var  numplayers : integer) ;

begin
  writeln ('Welcome to the game of Ludus Latrunculorum ');
  writeln ('  ****  Battle of Lysistrata  ****  ');
  writeln ;
 (* writeln (' How many players?   (enter 1 or 2) ');  *)
 (* read (numplayers);    *)
 numplayers := 2 ;
  repeat
    if numplayers = 1
      then oneplayer (side1)
      else if numplayers = 2
          then twoplayers (side1)
          else writeln ('wrong entry - enter 1 or 2 for number of players');
  until (numplayers = 1) or (numplayers = 2)
end ;

procedure setboard ( var board : boardtype ) ;

var
  row, col : integer ;

begin
  col:= 1 ;

  for row := 1 to 3 do
    begin
      while col <= 8 do
        begin
          board[row,col].full := true ;
          board[row,col].piece.morw :=woman ;
          board[row,col].piece.role := main ;
          board[row,col].piece.cancall := false ;
          col := col + 2 ;
        end ;
      if col = 9
        then col := 2
        else col := 1 ;
    end ;

  col:= 2;

  for row := 1 to 3 do
    begin
     while col <= 8 do
       begin
         board[row,col].full := true ;
         board[row,col].piece.cancall := true ;
         board[row,col].piece.morw := woman ;
         board[row,col].piece.role := chorus ;
         col := col + 2 ;
       end ;
     if col = 9
       then col:= 2
       else col:= 1 ;
   end ;
```

```
  for row := 4 to 5 do
      for col := 1 to 8 do
          board[row,col].full := false ;

  col := 1 ;

   for row := 6 to 8 do
    begin
      while col <= 8 do
       begin
          board[row,col].full := true ;
          board[row,col].piece.cancall := true ;
          board[row,col].piece.morw := man ;
          board[row,col].piece.role := chorus ;
          col := col + 2 ;
       end ;
      if col = 9
        then col := 2
        else col := 1 ;
  end ;

  col := 2 ;

  for row := 6 to 8 do
    begin
      while col <= 8 do
        begin
          board[row,col].full := true ;
          board[row,col].piece.morw := man ;
          board[row,col].piece.role := main ;
          board[row,col].piece.cancall := false ;
          col := col + 2 ;
       end ;
     if col = 9
      then col := 2
      else col := 1 ;
  end ;
end ;

function checkfull ( newrow, newcol : integer ;
                     board : boardtype ) : boolean ;

begin
  if board[newrow,newcol].full = true
    then checkfull := true
    else checkfull := false
end ;

function checkmove (board : boardtype ;
                    oldrow, oldcol, newrow, newcol : integer ;
                    side : char) : boolean ;

var
  wrongs, no_1, noempty, nofull, noblock,m1,m2,m3,m4,m5,m6 : string[80] ;
  rowab : integer ;
  colab : integer ;

begin
  textmode ;
  wrongs := 'You are trying to move the other side's piece -try again' ;
  no_1 :=  'Not moving in an l shape - try again.' ;
  noempty :=  'Can not move from an empty space - try again.' ;
  nofull := 'Can not move into already occupied space - try again.' ;
  noblock :='Can only move this piece one square at a time. try again!' ;
  m1 := 'Honestly, such behavior, with so much at stake ...' ;
```

```
  m3 := 'Unclean of that insolent lip !' ;
  m4 := 'Gross ineptitude. A sorry day for the Force ...' ;
  m5 := 'Such nanky-panky we have to thank for today's Utter Anarchy!' ;
  m6 := 'What-all's that bodacious ruckus?';
  m7 := 'Who-all's notion was this-hyer confabulation?' ;
  rowab := abs(oldrow - newrow) ;
  colab := abs(oldcol - newcol) ;
  checkmove := false ;
  if checkfull (oldrow, oldcol, board) = false
    then
      begin
        writeln (m1) ;
        writeln (noempty)
      end
  else if checkfull (newrow, newcol, board) = true
    then
      begin
        writeln (m2) ;
        writeln (nofull)
      end
  else if (board[oldrow,oldcol].piece.morw = man) and (side = 'w')
    then
      begin
        writeln (m3) ;
        writeln (wrongs)
      end
  else if (board[oldrow,oldcol].piece.morw = woman) and (side = 'm')
    then
      begin
        writeln (m4) ;
        writeln (wrongs)
      end
  else if board[oldrow,oldcol].piece.role = main
    then if ((rowab = 2) and (colab = 1)) or ((rowab = 1) and (colab = 2))
      then checkmove := true
      else
        begin
          writeln (m5) ;
          writeln (no_1)
        end
    else if ((rowab = 1) and (colab = 0)) or ((rowab = 0) and (colab = 1))
      then checkmove := true
      else
        begin
          writeln (m6) ;
          writeln (noblock);
        end
end ;

function menwon (board : boardtype) : boolean ;

var
  col, row : integer ;
  temp : boolean ;

begin
  row := 1 ;
  temp := true ;
  while (row <= 8) and (temp = true) do
    begin
      col := 1 ;
      while (col <= 8) and (temp = true) do
        begin
          if board[row,col].full = true
            then if board[row,col].piece.morw = woman
              then temp := false ;
```

```
          col := col + 1
        end ;
      row := row + 1
    end ;
  menwon := temp
end ;

function womenwon (board :boardtype ) : boolean ;

var
  col, row : integer ;
  temp : boolean ;

begin
  col := 1 ;
  temp := true ;
  while (col <= 8) and (temp = true) do
    begin
      row := 1;
      while (row <= 8) and (temp = true) do
        begin
          if board[row,col].full = true
            then if board[row,col].piece.morw = man
              then temp := false ;
          row := row + 1
        end ;
      row := row + 1
    end ;
  womenwon := temp
end ;

procedure godcall (    caller : char ;
                   var godcalled : godtype) ;

var   callnum : integer ;

begin
  fillscreen ( -1) ;
  smallsound ;
  randomize ;
  callnum := random (5) ;
  case callnum of
      0,2 : godcalled := ares ;
      1,3 : godcalled := aphrodite ;
      4   : godcalled := notthere ;
    end ;
end ;

procedure movepiece (var board : boardtype ;
                     oldrow, oldcol, newrow, newcol : integer) ;

begin
  board[newrow, newcol].full := true ;
  board[newrow, newcol].piece.morw := board[oldrow, oldcol].piece.morw ;
  board[newrow, newcol].piece.role := board[oldrow, oldcol].piece.role ;
  board[newrow, newcol].piece.cancall := board[oldrow, oldcol].piece.cancall ;
  board[oldrow, oldcol].full := false
end ;

procedure morecalls (     newcol : integer ;
                      var newsquare : square ;
                      var callcount : callarray ) ;

begin
  if newcol = 1
    then
```

```
      begin
        if newsquare.piece.morw = man
          then if newsquare.piece.cancall = true
            then
              begin
                callcount[2] := callcount[2] + 1 ;
                newsquare.piece.cancall := false ;
                writeln (' The men have another god-call');
                writeln (' for a total of ',callcount[2]:1) ;
              end
      end
    else if newcol = 8
      then if newsquare.piece.morw = woman
        then if newsquare.piece.cancall = true
          then
            begin
              callcount[1] := callcount[1] + 1 ;
              newsquare.piece.cancall := false ;
              writeln ('The women have another god-') ;
              writeln ('call for a total of', callcount[1]:1) ;
            end ;
end ;

procedure downcount (var callcount : callarray ;
                         caller : char ;
                     var noneleft : boolean) ;

const nocalls = 'you have no god-calls.' ;

begin
  noneleft := false ;
  case caller of
    'm'   : if callcount[2] = 0
              then
                begin
                  noneleft := true ;
                  writeln (nocalls)
                end
              else callcount[2] := callcount[2] - 1 ;
    'w'   : if callcount[1] = 0
              then
                begin
                  noneleft := true ;
                  writeln (nocalls)
                end
              else callcount[1] := callcount[1] - 1 ;
    end ;
  if noneleft = false
    then
      begin
        writeln (' Remaining god - calls :') ;
        writeln (' Women - ',callcount[1]:1,'  Men - ',callcount[2]:1) ;
      end ;
end ;

procedure takepieces (var board : boardtype ;
                          newrow, newcol : integer ;
                      var change : boolean) ;

var i, row : integer ;
    newmorw   : mwtype ;

begin
  change := false ;
  newmorw := board[newrow,newcol].piece.morw ;
  i := newrow - 1 ;
```

```
  while (board[i,newcol].full = true) and ((board[i,newcol].piece.morw = newmo
w) and (i > 1)) do
    i := i - 1 ;
  if (board[i,newcol].full = true) and ((board[i,newcol].piece.morw = newmorw) a
nd (i < newrow -1))
    then
      begin
        change := true ;
        for row:= (i + 1) to (newrow - 1) do
          board[row,newcol].full := false ;
      end ;
  i := newrow + 1 ;
  while (board[i,newcol].full = true) and ((board[i,newcol].piece.morw <> newmor
w) and (i < 8)) do
     i := i + 1 ;
  if (board[i,newcol].full = true) and (( board[i,newcol].piece.morw = newmorw)
and (i > newrow + 1))
    then
      begin
        change := true ;
        for row := (newrow + 1) to (i - 1) do
          board[row,newcol].full := false ;
      end ;
  i := newcol - 1 ;
  while board[newrow,i].full =true and (( board[newrow,i].piece.morw <> newmorw)
 and (i > 1 )) do
    i := i-1 ;
  if (board[newrow,i].full = true) and (( board[newrow,i].piece.morw = newmorw)
and (i < newcol-1  ) )
    then
      begin
        change := true ;
        for row := (i + 1) to (newcol - 1) do
          board[newrow,row].full := false ;
      end ;
  i := newcol + 1 ;
  while (board[newrow,i].full = true) and (( board[newrow,i].piece.morw <> newmo
rw) and (i < 8)) do
    i := i + 1 ;
  if (board[newrow,i].full = true) and (( board[newrow,i].piece.morw = newmorw)
and ( i > newcol + 1))
    then
      begin
        change := true ;
        for row := (newcol + 1) to (i - 1) do
          board[newrow,row].full := false ;
      end ;
end   ;

procedure drawboard (board : boardtype) ; (* dummy *)

const alllines = '  ----------------------------------------------------------
------' ;

var
  row, col : integer ;
  bsquare : square ;

begin
  clrscr ;
  writeln ('                                    column ') ;
  writeln ('       1       2       3       4       5       6       7       8') ;
  writeln (alllines) ;
  for row := 1 to 8 do
    begin
      for col := 1 to 8 do
```

```
        begin
          if col = 1
            then write ( row:1, ' |') ;
          bsquare := board[row,col] ;
          write ('  ') ;
          if bsquare.full = true
            then
              begin
                if bsquare.piece.morw = man
                  then write ('m-')
                  else write ('w-') ;
                if bsquare.piece.role = main
                  then write ('m  |')
                  else write ('c  |')
              end
            else write ('     |')
        end ;
      writeln ('|') ;
      writeln (alllines)
    end
end ;

procedure drawwc ;

var x1,x2,y1,y2,angle,color,radius : integer ;

begin
  x1 := 3 ;
  y1 := 6 ;
  radius := 3 ;
  color := 3 ;
  angle := 180 ;
  arc (x1,y1,angle, radius, color) ;
  x1 := 9 ;
  arc (x1, y1, angle, radius, color) ;
  x1 := 2 ;
  x2 := 8 ;
  y2 := 14 ;
  draw (x1, y1, x2, y2, color);
  x1 := 15 ;
  draw (x1, y1, x2, y2, color);
end ;

procedure drawwm ;

begin
  drawwc ;
  fillshape (9, 9, 3, 2) ;
end ;

procedure drawmc ;
begin
  draw (7, 2, 11, 2, 3) ;
  draw (7, 2, 7, 4, 3) ;
  draw (11, 2, 11, 4, 3) ;
  draw (3, 4, 7, 4, 3) ;
  draw (11, 4, 15, 4, 3) ;
  draw (3, 4, 3, 6, 3) ;
  draw (15, 4, 15, 6, 3) ;
  draw (3, 6, 7, 6, 3) ;
  draw (11, 6, 15, 6, 3) ;
  draw (7, 6, 9, 14, 3) ;
  draw (11, 6, 9, 14, 3) ;
end ;

procedure drawmm ;
```

```
begin
  drawmc ;
  fillshape (8, 3, 3, 2)
end ;


procedure drawboard2 (board : boardtype) ;

var wt, wb, wl, wr, row, col, color : integer ;
  bpiece : piecetype ;

begin
  palette (2) ;
  colortable (3,2,1,0) ;
  graphcolormode ;
  graphbackground (0) ;
  color := 3 ;
  writeln ('   1 2 3 4  5 6 7 8') ;
  writeln ;
  for row := 1 to 8 do
    begin
      wt := 16 + (16 * (row - 1)) ;
      wb := wt + 16 ;
      writeln ;
      writeln (row) ;
      for col := 1 to 8 do
        begin
          wl := 12 + (18 * (col - 1)) ;
          wr := wl + 18 ;
          graphwindow (wl,wt,wr,wb) ;
          draw (1, 1, 18, 1, color) ;
          draw (1, 1, 1, 16, color) ;
          draw (1, 16, 18, 16, color) ;
          draw (18, 1, 18, 16, color) ;
          if board [row,col].full = true
            then
              begin
                bpiece := board [row,col].piece ;
                if bpiece.morw = woman
                  then if bpiece.role = chorus
                    then drawwc
                    else drawwm
                  else if bpiece.role = chorus
                    then drawmc
                    else drawmm
              end ;
          graphwindow (0,0,319,199) ;
        end ;
    end ;
end  ;

procedure display ;

var cont : char ;

begin
  graphcolormode ;
  graphbackground (0) ;
  palette (1) ;
  colortable (3,2,1,0) ;
  writeln (' Heart shaped pieces are women.') ;
  writeln (' Swords are the men.') ;
  writeln (' Solid pieces are main actors -') ;
  writeln (' they move in an L - shape.') ;
  writeln (' Outlined pieces are the chorus -');
```

```
  writeln (' they move one square at a time.') ;
  graphwindow (20,120,40,160) ;
  crawwc ;
  graphwindow (60,120,80,160) ;
  drawwm ;
  graphwindow (100,120,120,160) ;
  drawmc ;
  graphwindow (140,120,160,160) ;
  drawmm ;
  writeln ('(Press any key to continue.)') ;
  cont := keyin ;
  textmode ;
end ;

procedure makechoice (    side : char ;
                       var action : char ;
                       var oldrow, oldcol, newrow, newcol : integer) ;

var oldsq, newsq : integer ;

begin
  writeln ;
  graphwindow (170,1,320,200) ;
  if side = 'm'
    then write ('Men - make your move.')
    else write (' Women - make your move.') ;
  writeln (' enter - m') ;
  writeln (' = move, q = quit, c = god-call') ;
  action := keyin ;
  write ('  -  ') ;
  if action = 'm'
    then
      begin
        writeln ('Enter old square then new') ;
        writeln ('square number.') ;
        readln (oldsq, newsq) ;
        oldrow := ((oldsq - 1) div 8) + 1 ;
        oldcol := ((oldsq - 1) mod 8) + 1 ;
        newrow := ((newsq - 1) div 8) + 1 ;
        newcol := ((newsq - 1) mod 8) + 1 ;
      end ;
end ;

procedure godtake (    godcalled : godtype ;
                    var board : boardtype) ;

var
  pieceout : boolean ;
  outrow, outcol, outsq : integer ;
  osquare : square ;

begin
  textmode ;
  if godcalled = notthere
    then
      begin
        writeln ('The gods are not at your beck and call - this is trivia.') ;
        writeln ('They have more fundamental business to engage in.') ;
        writeln ('Presently they are on a picnic.') ;
        delay (6000) ;
      end
  else if godcalled  = ares
    then
      begin
        writeln ('Ares has won the battle - the leader of the men should give th
e');
```

```
        writeln ('square number of the women's piece to be removed followed');
        writeln ('by enter.') ;
        delay (6000) ;
        repeat
          drawboard2 (board) ;
          writeln ;
          write ('Ares take your captive.') ;
          readln (outsq) ;
          outrow := ((outsq - 1) div 8) + 1 ;
          outcol := ((outsq - 1) mod 8) + 1 ;
          osquare := board[outrow, outcol] ;
          if (osquare.full = true) and (osquare.piece.morw = woman)
            then
              begin
                pieceout := true ;
                board[outrow, outcol].full := false
              end
            else
              begin
                pieceout := false ;
                writeln ('There is no women"s piece');
                writeln ('in that square - try again.')
              end ;
        until pieceout = true
      end
    else
      begin
        writeln ('Aphrodite has won - the women should enter the square number')
;
        writeln (' of the men"s piece to be removed followed by enter. ') ;
        delay (6000) ;
        repeat
          drawboard2 (board) ;
          writeln ;
          write ('Aphrodite take a captive.') ;
          readln (outrow, outcol) ;
          osquare := board[outrow, outcol] ;
          if (osquare.full = true) and (osquare.piece.morw = man)
            then
              begin
                pieceout := true ;
                board[outrow, outcol].full := false
              end
            else
              begin
                pieceout := false ;
                writeln ('There is no men"s piece');
                writeln ('to remove there - try again.')
              end ;
        until pieceout = true
      end  ;
end ;

procedure switchsides (var side : char) ;

begin
  if side = 'm'
    then side := 'w'
    else side := 'm'
end ;

procedure compmove (var board : boardtype ;  (* dummy *)
                        side : char) ;

begin
end ;
```

```
(*****************************************************************************)
(*                              main program                                 *)
(*****************************************************************************)

begin
  textmode(c80) ;
  moveset := ['m','M'] ;
  outloop1 := ['Y','y','M','m'] ;
  outloop2 := ['Y','y'] ;
  repeat
    writestart ;
    display ;
    callcount[1] := 3 ;
    callcount[2] := 3 ;
    setboard (board) ;
    getnum (side, numplayers) ;
    repeat
      godcall (side, godcalled) ;
    until godcalled <> notthere ;
    if godcalled = ares
      then
        begin
          writeln ('Ares won the first battle, the men will start the game.') ;
          side := 'm'
        end
      else
        begin
          writeln ('Aphrodite won the first battle, the women will start the gam
e.') ;
          side := 'w'
        end ;
    delay (6000) ;
    newboard := 'n' ;
    repeat
      repeat
        if (newboard ='n') or (action = 'n')
          then
            begin
              newboard := 'a' ;
              drawboard2 (board) ;
            end ;
        makechoice (side, action, oldrow, oldcol, newrow, newcol) ;
        if action = 'c'
          then
            begin
              downcount (callcount, side, noneleft) ;
              if noneleft = false
                then
                  begin
                    godcall (side, godcalled) ;
                    godtake (godcalled, board);
                    newboard := 'n' ;
                  end ;
            end
        else if action = 'q'
          then
            begin
              textmode ;
              writeln ('Are you sure that you want to quit - type y or n');
              action := keyin ;
              if action in outloop2
                then goto quit ;
            end
        else if action = 'm'
          then if checkmove (board, oldrow, oldcol, newrow, newcol, side) = true
            then
```

```
                begin
                  movepiece (board, oldrow, oldcol, newrow, newcol) ;
                end
              else action := 'n';
          delay (2500)
        until action in outloop1 ;
        drawboard2 (board);
        takepieces (board, newrow, newcol, change) ;
        if change = true
          then
            begin
              writeln ;
              writeln ('press any key to next board.') ;
              newboard := keyin ;
              newboard := 'n';
            end ;
        if numplayers = 2
          then switchsides (side)
          else compmove (board, side) ;
      until menwon (board) or womenwon (board) ;
      if not (action in outloop2)
        then drawboard2 (board) ;
      if menwon (board)
        then writeln ('Ares and the men of Greece win the game.')
        else if womenwon (board)
          then
            begin
              writeln ('Aphrodite and the women led') ;
              writeln ('by Lysistrata have won the game.') ;
            end ;
quit: textmode ;
      writeln (' Type the letter n if you want to start a new game.') ;
      writeln (' Any other key will exit the program.') ;
      newgame := keyin ;
  until newgame <> 'n'
end.
```