```
program oxo(input, output);

#include "status.p"
#include "sqvals.p"
#include "play.p"
#include "gamestate.p"

var
    c: char;
    sq: integer;
    human, computer: OXO;
begin
    initboard(9);
    initlines;
    display3x3;

    write('Do you want to play X or O?');
    repeat
        readln(c)
    until (c = 'O') or (c = 'o') or (c = 'X') or (c = 'x');
    if (c = 'O') or (c = 'o') then begin
        human := O;
        computer := X
    end else begin
        human := X;
        computer := O
    end;

    write('Who is to start (X or O)?');
    repeat
        readln(c)
    until (c = 'O') or (c = 'o') or (c = 'X') or (c = 'x');
    if (c = 'O') or (c = 'o') then
        startplayer := O
    else
        startplayer := X;

    if startplayer <> human then begin
        player := computer;
        put(computer, maxindex);
        display3x3
    end;

    write('Your turn (1-9 or Q to quit) > ');
    readln(c);
    while (c <> 'Q') and (c <> 'q') do begin
        sq := ord(c) - ord('0');
        square := sq - 1;
        if (sq < 1) or (sq > 9) or not availsquare then
            writeln('illegel move, try again.')
        else begin
            if full then
                writeln('can''t put more in')
            else
                put(human, sq - 1);
            display3x3;
            if owon then begin
                writeln('O wins!');
                clearall;
                display3x3
            end else if xwon then begin
                writeln('X wins!');
                clearall;
                display3x3
            end else if draw then begin
                writeln('Draw!');
```

```
                clearall;
                display3x3
            end;
            writeln('computer plays ', maxindex + 1: 1);
            put(computer, maxindex);
            display3x3;
            if owon then begin
                writeln('O wins!');
                clearall;
                display3x3
            end else if xwon then begin
                writeln('X wins!');
                clearall;
                display3x3
            end else if draw then begin
                writeln('Draw!');
                clearall;
                display3x3
            end
        end;
        write('Your turn (1-9 or Q to quit) > ');
        readln(c)
    end;
    writeln('Bye.')
end. { oxo }
```

```
{ display3x3.p - display a 3x3 oxo board }

procedure display3x3;
var
    i, j: integer;
begin
    for i := 0 to 2 do begin
        if i > 0 then
            writeln('-+-+-');
        for j := 0 to 2 do begin
            if j > 0 then
                write('|');
            case board[i * 3 + j] of
                O:
                    write('O');
                X:
                    write('X');
                otherwise
                    write(i * 3 + j + 1: 1)
            end
        end;
        writeln
    end
end; { display3x3 }
```

```
program test1;

#include "position.p"


begin
    initboard(9);
    put(X, 1);
    put(O, 8);
    display3x3;
    writeln;
    put(X, 3);
    clear(1);
    display3x3;
    writeln;
    clearall;
    display3x3
end. { test1 }
```

```
program test2;

#include "status.p"


var
    c: char;
    sq: integer;
begin
    initboard(9);
    initlines;
    display3x3;
    write('Input (Q to quit) > ');
    read(c);
    while (c <> 'Q') and (c <> 'q') do begin
        readln(sq);
        if full then
            writeln('can''t put more in')
        else if (c = 'X') or (c = 'x') then
            put(X, sq - 1)
        else if (c = 'O') or (c = 'o') then
            put(O, sq - 1)
        else
            clear(sq - 1);
        display3x3;
        if owon then begin
            writeln('O wins!');
            clearall;
            display3x3
        end else if xwon then begin
            writeln('X wins!');
            clearall;
            display3x3
        end else if draw then begin
            writeln('Draw!');
            clearall;
            display3x3
        end;
        write('Input (Q to quit) > ');
        read(c)
    end;
    writeln('Bye.')
end. { test2 }
```

```
var
    startplayer: OXO;

function end_of_game: boolean;
begin
    end_of_game := xwon or owon or draw
end; { end_of_game }


function x_to_play: boolean;
begin
    x_to_play := not end_of_game and
        ((startplayer = X) and (nofo = nofx)) or (nofo > nofx)
end; { x_to_play }


function o_to_play: boolean;
begin
    o_to_play := not end_of_game and
        ((startplayer = O) and (nofo = nofx)) or (nofx > nofo)
end; { o_to_play }
```

```
#include "position.p"

const

    MAXLINES = 76;

type
    LINE = set of SQUARE;
    LINES =
        record
            noflines: integer;
            line: array [0..MAXLINES - 1] of LINE
        end; { should be set of LINE }

var
    lines: LINES;

function isline(l: LINE): boolean;
var
    i: integer;
    result: boolean;
begin
    result := false;
    i := 0;
    while result = false and (i < lines.noflines) do begin
        if l = lines.line[i] then
            result := true;
        i := i + 1
    end;
    isline := result
end; { isline }


procedure initlines;
begin
    lines.noflines := 8;
    lines.line[0] := [0, 1, 2];
    lines.line[1] := [3, 4, 5];
    lines.line[2] := [6, 7, 8];
    lines.line[3] := [0, 3, 6];
    lines.line[4] := [1, 4, 7];
    lines.line[5] := [2, 5, 8];
    lines.line[6] := [0, 4, 8];
    lines.line[7] := [2, 4, 6]
end; { initlines }
```

```
function findmaxindex: SQUARE;
var
    maxval, foundindex: SQUARE;
    looping: boolean;
begin
    square := 0;                                { global variable }
    while not availsquare do
        square := square + 1;
    maxval := 0;
    foundindex := square;
    looping := true;
    while looping do begin
        if availsquare and (cursqval >= maxval) then begin
            maxval := cursqval;
            foundindex := square
        end;
        if square < nofsquares then
            square := square + 1
        else if square = nofsquares then
            looping := false
    end;
    findmaxindex := foundindex
end; { findmaxindex }


function maxindex: SQUARE;
begin
    maxindex := findmaxindex
end; { maxindex }
```

```
const
    MAXSQS = 64;

type
    SQUARE = 0..MAXSQS - 1;
    OXO = (O, X, U);
    POSITION = array [0..MAXSQS - 1] of OXO;
    BOARD = set of SQUARE;

var
    board: POSITION;
    nofsquares: integer;

procedure put(item: OXO; p: integer);
begin
    board[p] := item
end; { put }


procedure clear(p: integer);
begin
    board[p] := U
end; { clear }


procedure clearall;
var
    i: integer;
begin
    for i := 0 to nofsquares - 1 do
        clear(i)
end; { clearall }


procedure initboard(size: integer);
begin
    nofsquares := size;
    clearall
end; { initboard }

#include "display3x3.p"
```

```
type
    VALUE =
        record
            Os, Xs, Us: integer
        end;

var
    square: SQUARE;
    player: OXO;

function availsquare: boolean;
begin
    availsquare := board[square] = U
end; { availsquare }


function linval(l: LINE): VALUE;
var
    i: integer;
    result: VALUE;
begin
    result.Xs := 0;
    result.Os := 0;
    result.Us := 0;
    for i := 0 to nofsquares - 1 do
        if i in l then
            case board[i] of
                X:
                    result.Xs := result.Xs + 1;
                O:
                    result.Os := result.Os + 1;
                otherwise
                    result.Us := result.Us + 1
            end;
    linval := result
end; { linval }


function weighting(v: VALUE; p: OXO): integer;
var
    players, opponents: integer;
begin
    if p = X then begin
        players := v.Xs;
        opponents := v.Os
    end else begin
        players := v.Os;
        opponents := v.Xs
    end;
    if (players > 0) and (opponents > 0) then
        weighting := -1
    else if (opponents = 0) and (v.Us = 1) then
        weighting := 100
    else if (players = 0) and (v.Us = 1) then
        weighting := 20
    else if opponents = 0 then
        weighting := 3
    else if players = 0 then
        weighting := 1
    else
        weighting := 0
end; { weighting }


function sqval(lns: LINES): integer;
var
```

```
    i: integer;
    result: integer;
begin
    result := 0;
    for i := 0 to lns.noflines - 1 do
        result := result + weighting(linval(lns.line[i]), player);
    sqval := result
end; { sqval }


function linesthru(s: SQUARE): LINES;
var
    i: integer;
    result: LINES;
begin
    result.noflines := 0;
    for i := 0 to lines.noflines - 1 do
        if s in lines.line[i] then begin
            result.line[result.noflines] := lines.line[i];
            result.noflines := result.noflines + 1
        end;
    linesthru := result
end; { linesthru }


function cursqval: integer;
begin
    cursqval := sqval(linesthru(square))
end; { cursqval }
```

```
#include "line.p"
function nofx: integer;

var
    i: integer;
    result: integer;
begin
    result := 0;
    for i := 0 to nofsquares - 1 do
        if board[i] = X then
            result := result + 1;
    nofx := result
end; { nofx }


function nofo: integer;
var
    i: integer;
    result: integer;
begin
    result := 0;
    for i := 0 to nofsquares - 1 do
        if board[i] = O then
            result := result + 1;
    nofo := result
end; { nofo }


function full: boolean;
begin
    full := nofo + nofx = nofsquares
end; { full }


function xwon: boolean;
var
    i: integer;
    result: boolean;
    setx: set of SQUARE;
begin
    setx := [];
    for i := 0 to nofsquares - 1 do
        if board[i] = X then
            setx := setx + [i];
    i := 0;
    result := false;
    while (result = false) and (i < lines.noflines) do begin
        if lines.line[i] <= setx then
            result := true;
        i := i + 1
    end;
    xwon := result
end; { xwon }


function owon: boolean;
var
    i: integer;
    result: boolean;
    seto: set of SQUARE;
begin
    seto := [];
    for i := 0 to nofsquares - 1 do
        if board[i] = O then
            seto := seto + [i];
    i := 0;
```

```
    result := false;
    while (result = false) and (i < lines.noflines) do begin
        if lines.line[i] <= seto then
            result := true;
        i := i + 1
    end;
    owon := result
end; { owon }


function draw: boolean;
begin
    draw := not xwon and not owon and full
end; { draw }
```