Program AQAReverse;

{Skeleton Program code for the AQA COMP1 Summer 2016 examination

this code whould be used in conjunction with the Preliminary Material

written by the AQA COMP1 Programmer Team

developed in the Turbo Pascal 7 programming environment}

{Centres may add the SysUtils library if their version of Pascal supports this}

{Permission to make this change to the Skeleton Program does not

need to be obtained from AQA/AQA Programmer - just remove the braces around Uses SysUtils;}

{Uses

SysUtils;}

Type

TBoard = Array[1..9, 1..9] Of Char;

Var

Choice : Char;

PlayerName : String;

BoardSize : Integer;

Procedure SetUpGameBoard(Var Board : TBoard; BoardSize : Integer);

Var

Row : Integer;

Column : Integer;

Begin

For Row := 1 To BoardSize

Do

For Column := 1 To BoardSize

Do

If (Row = (BoardSize + 1) Div 2) And (Column = (BoardSize + 1) Div 2 + 1)

Or (Column = (BoardSize + 1) Div 2) And (Row = (BoardSize + 1) Div 2 + 1)

Then Board[Row, Column] := 'C'

Else

If (Row = (BoardSize + 1) Div 2 + 1) And (Column = (BoardSize + 1) Div 2 + 1)

Or (Column = (BoardSize + 1) Div 2) And (Row = (BoardSize + 1) Div 2)

Then Board[Row, Column] := 'H'

Else Board[Row, Column] := ' '

End;

Function ChangeBoardSize : Integer;

Var

BoardSize : Integer;

Begin

Repeat

Write('Enter a board size (between 4 and 9): ');

Readln(BoardSize);

Until (BoardSize >= 4) And (BoardSize <= 9);

ChangeBoardSize := BoardSize;

End;

Function GetHumanPlayerMove(PlayerName : String) : Integer;

Var

Coordinates : Integer;

Begin

Write(PlayerName, ' enter the coordinates of the square where you want to place your piece: ');

Readln(Coordinates);

GetHumanPlayerMove := Coordinates;

End;

Function GetComputerPlayerMove(BoardSize : Integer) : Integer;

Begin

GetComputerPlayerMove := (Random(BoardSize) + 1) \* 10 + Random(BoardSize) + 1;

End;

Function GameOver(Board : TBoard; BoardSize : Integer) : Boolean;

Var

Row : Integer;

Column : Integer;

Begin

GameOver := True;

For Row := 1 To BoardSize

Do

For Column := 1 To BoardSize

Do

If Board[Row, Column] = ' '

Then GameOver := False;

End;

Function GetPlayersName : String;

Var

PlayerName : String;

Begin

Write('What is your name? ');

Readln(PlayerName);

GetPlayersName := PlayerName;

End;

Function CheckIfMoveIsValid(Board : TBoard; Move : Integer) : Boolean;

Var

Row : Integer;

Column : Integer;

MoveIsValid : Boolean;

Begin

Row := Move Mod 10;

Column := Move Div 10;

MoveIsValid := False;

If Board[Row, Column] = ' '

Then MoveIsValid := True;

CheckIfMoveIsValid := MoveIsValid;

End;

Function GetPlayerScore(Board : TBoard; BoardSize : Integer; Piece : Char) : Integer;

Var

Score : Integer;

Row : Integer;

Column : Integer;

Begin

Score := 0;

For Row := 1 To BoardSize

Do

For Column := 1 To BoardSize

Do

If Board[Row, Column] = Piece

Then Score := Score + 1;

GetPlayerScore := Score;

End;

Function CheckIfThereArePiecesToFlip(Board : TBoard;

BoardSize, StartRow, StartColumn, RowDirection, ColumnDirection : Integer) : Boolean;

Var

RowCount : Integer;

ColumnCount : Integer;

FlipStillPossible : Boolean;

FlipFound : Boolean;

OpponentPieceFound : Boolean;

Begin

RowCount := StartRow + RowDirection;

ColumnCount := StartColumn + ColumnDirection;

FlipStillPossible := True;

FlipFound := False;

OpponentPieceFound := False;

While (RowCount <= BoardSize) And (RowCount >= 1) And (ColumnCount >= 1)

And (ColumnCount <= BoardSize) And FlipStillPossible And Not FlipFound

Do

Begin

If Board[RowCount, ColumnCount] = ' '

Then FlipStillPossible := False

Else

If Board[RowCount, ColumnCount] <> Board[StartRow, StartColumn]

Then OpponentPieceFound := True

Else

If (Board[RowCount, ColumnCount] = Board[StartRow, StartColumn]) And Not OpponentPieceFound

Then FlipStillPossible := False

Else FlipFound := True;

RowCount := RowCount + RowDirection;

ColumnCount := ColumnCount + ColumnDirection;

End;

CheckIfThereArePiecesToFlip := FlipFound;

End;

Procedure FlipOpponentPiecesInOneDirection(Var Board : TBoard;

BoardSize, StartRow, StartColumn, RowDirection, ColumnDirection : Integer);

Var

RowCount : Integer;

ColumnCount : Integer;

FlipFound : Boolean;

Begin

FlipFound := CheckIfThereArePiecesToFlip(Board, BoardSize, StartRow, StartColumn, RowDirection, ColumnDirection);

If FlipFound

Then

Begin

RowCount := StartRow + RowDirection;

ColumnCount := StartColumn + ColumnDirection;

While (Board[RowCount, ColumnCount] <> ' ') And (Board[RowCount, ColumnCount] <> Board[StartRow, StartColumn])

Do

Begin

If Board[RowCount, ColumnCount] = 'H'

Then Board[RowCount, ColumnCount] := 'C'

Else Board[RowCount, ColumnCount] := 'H';

RowCount := RowCount + RowDirection;

ColumnCount := ColumnCount + ColumnDirection;

End;

End;

End;

Procedure MakeMove(Var Board : TBoard; BoardSize, Move : Integer; HumanPlayersTurn : Boolean);

Var

Row : Integer;

Column : Integer;

Begin

Row := Move Mod 10;

Column := Move Div 10;

If HumanPlayersTurn

Then Board[Row, Column] := 'H'

Else Board[Row, Column] := 'C';

FlipOpponentPiecesInOneDirection(Board, BoardSize, Row, Column, 1, 0);

FlipOpponentPiecesInOneDirection(Board, BoardSize, Row, Column, -1, 0);

FlipOpponentPiecesInOneDirection(Board, BoardSize, Row, Column, 0, 1);

FlipOpponentPiecesInOneDirection(Board, BoardSize, Row, Column, 0, -1);

End;

Procedure PrintLine(BoardSize : Integer);

Var

Count : Integer;

Begin

Write(' ');

For Count := 1 To BoardSize \* 2 - 1

Do Write('\_');

Writeln;

End;

Procedure DisplayGameBoard(Board : TBoard; BoardSize : Integer);

Var

Row : Integer;

Column : Integer;

Begin

Writeln;

Write(' ');

For Column := 1 To BoardSize

Do

Begin

Write(' ');

Write(Column);

End;

Writeln;

PrintLine(BoardSize);

For Row := 1 To BoardSize

Do

Begin

Write(Row);

Write(' ');

For Column := 1 To BoardSize

Do

Begin

Write('|');

Write(Board[Row, Column]);

End;

Writeln('|');

PrintLine(BoardSize);

Writeln;

End;

End;

Procedure DisplayMenu;

Begin

Writeln('(p)lay game');

Writeln('(e)nter name');

Writeln('(c)hange board size');

Writeln('(q)uit');

Writeln;

End;

Function GetMenuChoice(PlayerName : String) : Char;

Var

Choice : Char;

Begin

Write(PlayerName, ' enter the letter of your chosen option: ');

Readln(Choice);

GetMenuChoice := Choice;

End;

Procedure PlayGame(PlayerName : String; BoardSize : Integer);

Var

Board : TBoard;

HumanPlayersTurn : Boolean;

Move : Integer;

HumanPlayerScore : Integer;

ComputerPlayerScore : Integer;

MoveIsValid : Boolean;

Begin

SetUpGameBoard(Board, BoardSize);

HumanPlayersTurn := False;

Repeat

HumanPlayersTurn := Not HumanPlayersTurn;

DisplayGameBoard(Board, BoardSize);

MoveIsValid := False;

Repeat

If HumanPlayersTurn

Then Move := GetHumanPlayerMove(PlayerName)

Else Move := GetComputerPlayerMove(BoardSize);

MoveIsValid := CheckIfMoveIsValid(Board, Move);

Until MoveIsValid;

If Not HumanPlayersTurn

Then

Begin

Writeln('Press the Enter key and the computer will make its move');

Readln;

End;

MakeMove(Board, BoardSize, Move, HumanPlayersTurn);

Until GameOver(Board, BoardSize);

DisplayGameBoard(Board, BoardSize);

HumanPlayerScore := GetPlayerScore(Board, BoardSize, 'H');

ComputerPlayerScore := GetPlayerScore(Board, BoardSize, 'C');

If HumanPlayerScore > ComputerPlayerScore

Then Writeln('Well done, ', PlayerName, ', you have won the game!')

Else

If HumanPlayerScore = ComputerPlayerScore

Then Writeln('That was a draw!')

Else

Begin

Writeln('The computer has won the game!');

End;

Writeln;

End;

Begin

Randomize;

BoardSize := 6;

PlayerName := '';

Repeat

DisplayMenu;

Choice := GetMenuChoice(PlayerName);

Case Choice Of

'p' : PlayGame(PlayerName, BoardSize);

'e' : PlayerName := GetPlayersName;

'c' : BoardSize := ChangeBoardSize;

End;

Until Choice = 'q';

End.