#include <iostream>

#include <conio.h>

#include <string>

using namespace std;

void makeHumanMove(string);

int aiLogic();

void showBoard();

bool checkForWin(char token);

bool CheckHumanMoveIsAvailible(string human);

//2d Array board

char boardArray[][3] = { { '1', '2', '3' }, { '4', '5', '6' }, { '7', '8', '9' } };

int main()

{

string humanMove, aiMove;//declaring string for user input, easier for input

//intro to game

cout << "Choose your selection by entering 1-9 " << endl;

cout << "Human will be X" << endl;

cout << "Cpu will be O" << endl;

showBoard();//showing the board

//for loop for allopwing human and IA to take a turn

for (int turn = 1; turn <= 5; turn++)

{

humanMove = "null"; // human hasn't made move yet..

// while loop for validation

while (humanMove == "null")

{

cout << "Pick a number (1-9)." << endl;//telling human to pick a number

cin >> humanMove;//inputs my string

if (!CheckHumanMoveIsAvailible(humanMove)) //not a human move that is avaiable

{

cout << "That move is not availible!." << endl;//telling human to pick a number

humanMove = "null";//if move is not valid, set the move set the move back to null

} // so that the loop can prompt and check again

}

makeHumanMove(humanMove);

if (checkForWin('X'))//if statement checking to see if the human move made above was a winning move.

{

showBoard();//showing the board

cout << "X wins." << endl;//stating winner

break;// exits loop if winner is declared

}

// same logic as above but for IA

aiLogic();

if (checkForWin('O'))

{

showBoard();

cout << "O wins." << endl;

break;//leaves logic

}

//show board to continue game

showBoard();

}

cout << "Game over." << endl;//displays "game over" after winner

\_getch();

return 0;

}//end main

void showBoard()//method that allows board to show

{

//array displaying

for (int col = 0; col < 3; col++)

{

for (int row = 0; row < 3; row++)

{

cout << boardArray[col][row] << " ";

}//end row

cout << endl;

}//end col

cout << endl;

}

bool CheckHumanMoveIsAvailible(string human)

{

if (human == "1" && boardArray[0][0] != 'X' && boardArray[0][0] != 'O')

{

return true;

}

if (human == "2" && boardArray[0][1] != 'X' && boardArray[0][1] != 'O')

{

return true;

}

if (human == "3" && boardArray[0][2] != 'X' && boardArray[0][2] != 'O')

{

return true;

}

if (human == "4" && boardArray[1][0] != 'X' && boardArray[1][0] != 'O')

{

return true;

}

if (human == "5" && boardArray[1][1] != 'X' && boardArray[1][1] != 'O')

{

return true;

}

if (human == "6" && boardArray[1][2] != 'X' && boardArray[1][2] != 'O')

{

return true;

}

if (human == "7" && boardArray[2][0] != 'X' && boardArray[2][0] != 'O')

{

return true;

}

if (human == "8" && boardArray[2][1] != 'X' && boardArray[2][1] != 'O')

{

return true;

}

if (human == "9" && boardArray[2][2] != 'X' && boardArray[2][2] != 'O')

{

return true;

}

return false;

}//end void

//method allowing me to set my move

void makeHumanMove(string human)

{

if (human == "1")

boardArray[0][0] = 'X';

else if (human == "2")

boardArray[0][1] = 'X';

else if (human == "3")

boardArray[0][2] = 'X';

else if (human == "4")

boardArray[1][0] = 'X';

else if (human == "5")

boardArray[1][1] = 'X';

else if (human == "6")

boardArray[1][2] = 'X';

else if (human == "7")

boardArray[2][0] = 'X';

else if (human == "8")

boardArray[2][1] = 'X';

else if (human == "9")

boardArray[2][2] = 'X';

}//end void makeMove

//

int aiLogic()

{

for (int col = 0; col < 3; col++)

{

for (int row = 0; row < 3; row++)

{

if (boardArray[col][row] != 'X' && boardArray[col][row] != 'O')//checking space to see if available

{

boardArray[col][row] = 'O';//if available, take space

return 1;//this return exits this method if IA has taken a space

}

}//end row

}//end col

}

bool checkForWin(char token)//checking for win

{

for (int counter = 0; counter < 3; counter++)//looping through columns and rows at sametime for wins

{

if (boardArray[counter][0] == token && boardArray[counter][1] == token && boardArray[counter][2] == token)

{

return true;

}

if (boardArray[0][counter] == token && boardArray[1][counter] == token && boardArray[2][counter] == token)

{

return true;

}

}

// checking the diagonals

if (boardArray[0][0] == token && boardArray[1][1] == token && boardArray[2][2] == token)

{

return true;

}

if (boardArray[0][2] == token && boardArray[1][1] == token && boardArray[2][0] == token)

{

return true;

}

return false;//returns if no winner

}