Odd Semester (2020)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work****)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Mikha** | | **Student ID Number**  **2001585575** | |
| 1. | | **Putri Yupikha** |  | |
|  |  |
| **Course Code** | **: COMP6335** |  |  | | **Course Name** | | **: Introduction to Programming** | |
| **Class** | **: L1AC** |  |  | | **Name of Lecturer(s)** | | **:** 1. Bagus Kerthyayana | |
|  |  |  |  | |  | | 2. Stavin | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment**  (if any) | : Interest and Depreciation | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 6-11-2016** |  |  | | **Submission Date** | | **: 6-11-2016** | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Mikha Putri Yupikha

**Table of Contents**

**SECTION 1: Description……………………………………………………..... 3**

**SECTION 2: Design………………………………………………………………. 5**

**2A: Plan……………………………………………………………………… 5**

**2B: Function Explanations…………………………………………. 6**

**SECTION 3: Reflections……………………………………………………….. 8**

**3A: Lessons Learned………………………………………………….. 8**

**3B: Problems……………………………………………………………. 10**

**SECTION 4: Source Code……………………………………………………. 11**

**SECTION 1: DESCRIPTION**

Tic-Tac-Toe was first played in the Roman Empire. They called it as Terni Lapilli. As time passed by the world knows it as Tic-Tac-Toe. It is a simple game for two-players in which they take turns to mark their chosen number in 3x3 board game. The objective of this game is to have three marks in line (horizontally, vertically or diagonally).

Player 1 will be marked as ‘X’

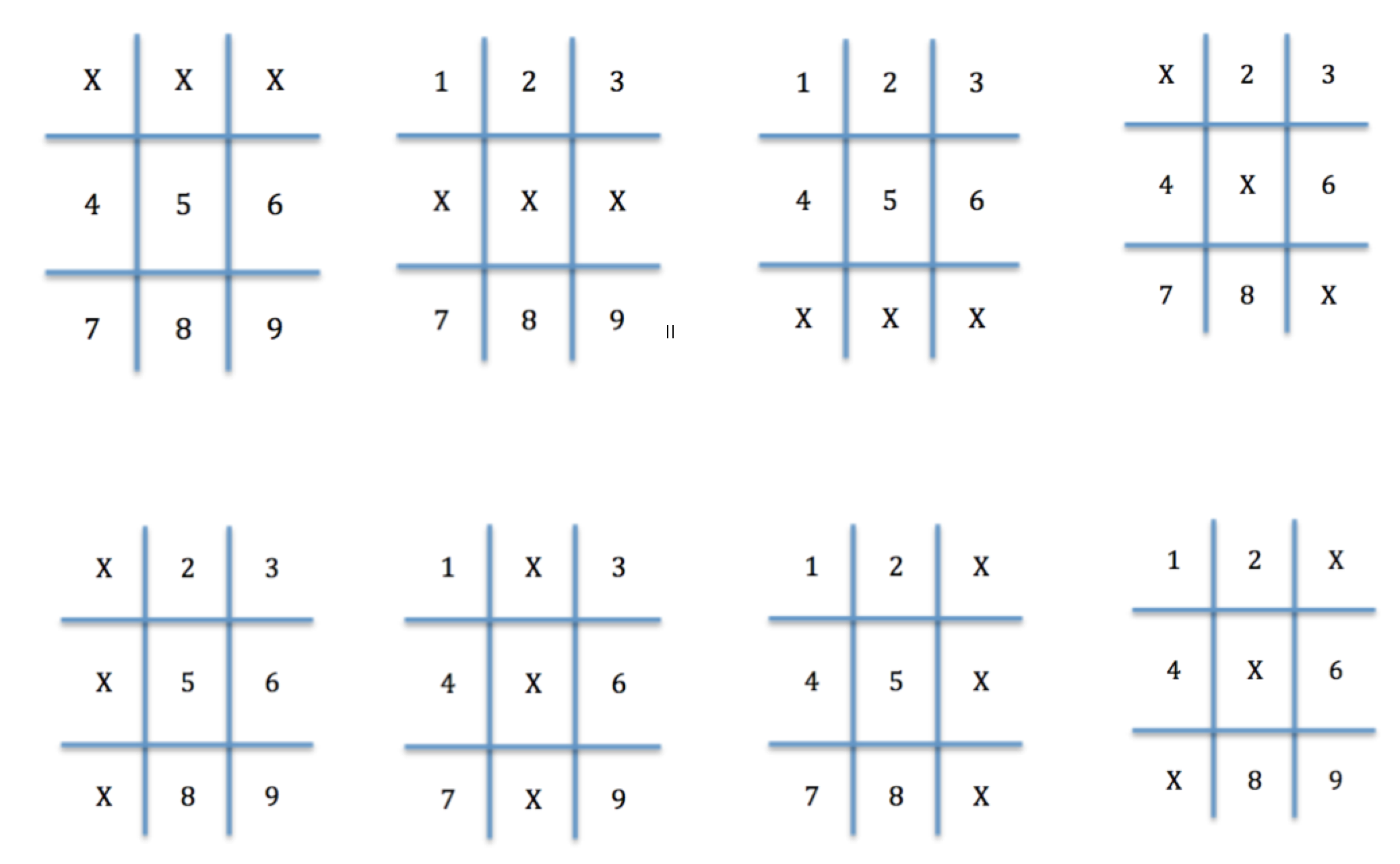
Player 2 will be marked as ‘O’

* Board Display:

1 2 3

4 5 6

7 8 9

* The Objective:
* Can be marked with ‘X’ for Player1 or ‘O’ for Player2.
* Process of The Game:

1. First players choose number between 1-9. The chosen number will be marked as ‘X’.
2. Next, second player choose number between 1-9 but cannot be the same number as the Player 1 before.
3. Do it alternately until one of the player win by making the marks in order (horizontally, vertically, or diagonally).
4. If the board already full and none of the players achieve the goal, the game is draw.

**SECTION 2: DESIGN**

**2A. PLAN**

* Hierarchy Chart

Board Layout

*(void boardGame)*

Decide to play again or quit

*(bool playAgain)*

Check if win/decide the winner

*(bool check)*

Check if draw

*(bool isBoardFilled)*

Game flow

*(void gameplay (int playerPlay))*

**Tic Tac Toe**

**Game**

***(class TIcTacToe)***

Instruction

*(void instruction)*

Welcome Message / Menu

*(bool menu)*

**Board Game**

**Header**

**Process**

**2B. FUNCTION EXPLANATIONS**

* main Function

This function is to combine all the functions I made and most of them run in the main. Some function like *menu* and *instructions* are declared here. Also, welcoming message of the program.

* class TicTacToe

This class stores the board number of the board game and functions to support this program works.

* Private member:
* char boardNumber [9] = {'1','2','3','4','5','6','7','8','9'};
* Public members:
* void boardGame();
* bool check();
* bool isBoardFilled();
* void gamePlay(int playerPlay);
* bool playAgain();
* boardGame Function

The use of this function is to display the layout of the 3 x 3 board game. It holds *cout* function to output ‘-‘ and ‘|’.

* check Function

This function has data type Boolean. It checks whether one of the player is win by checking the board number using *if* function. While the game is going on, it keeps checking whether one of the players already reaches the objective. If it is, then it *return true*. If not, *return false*.

* isBoardFilled Function

The purpose of this function is for checking if the game is draw / no winners. It has Boolean data type. First, I’m assuming that the board is already filled (*bool filled = true).*  By using *for loop,* it checks the board one by one if it is filled with mark or not. If the one of board number has not been filled with ‘X’ or ‘O’ then my assumption is wrong {*filled = false).*

* gameplay Function

Basically, this is how the overall game will flow. It has *void*  data type and *while loop* . The loop will keep going if the game is not draw (*boardFilled = false)*  and there is no winner yet (*i = false).* Before the loop, I declare the first player is 1, i (to check winner) is false and boardfilled is false. Inside the while loop, I decide the player, whether it is Player1 or Player2. Then I display the board by using the *boardGame* function. After that I decide that Player1 has ‘X’ mark and Player2 has ‘O’ mark using *if* function.

Then, I ask the player to enter a number between 1 – 9. I use  *if*  function to mark the board number. For instance, the player choose number 1, then *boardNumber[0]*  is marked. If the player input number outside the range, the game will be stopped and ask if they want to replay the game or not.

I use *playerPlay++* to alternate players. Every loop, the program will check if the game is draw (*boardFilled = isBoardFilled()*) and check winner (*I = check()*). If it is false, the loop will start again but if it is true it will stop the loop. If the loop stopped because there is a winner, it will display which player wins or else the game is draw.

* playAgain Function

This function has Boolean data type. I assume that the start is true (*bool start = true*). Then, I ask whether the player wants to repeat the game or not. They should enter ‘Y’ / ‘y’ for yes or ‘N’ / ‘n’ for no. Using the *if* function, if yes, the program will go directly to *menu* function. If no, the program will terminate. I put validation if the player input other than these character.

* menu Function

The intention of this function is mostly to display the menu. It has Boolean data type and *while* loop. This function will be the very first to show in the program. There are 3 options. The first option is to play the game, the second option is to read the instruction and the last option is to quit the program. If the player chooses option 2, they are able to get back to the menu and play or quit. I also put validation, if the player chooses number outside the range, error message will come out.

* instruction Function

This function only show the general description of the game and steps to play the game.

* UML Diagram

|  |
| --- |
| TicTacToe |
| **-**boardNumber[9] |
| +boardGame: void |
| +check: bool |
| +isBoardFilled: bool  +gamePlay: void (playerPlay: int)  +playAgain: bool |

**SECTION 3: REFLECTION**

**3A. LESSONS LEARNED**

* Journal
* October 6th

On this date I started to create the description and the process of the game. I planned the display of the game, how the game will look like. I learned how to make hierarchy chart. Next update, I will start to code the display and think the winner criteria.

* October 13th

On this update, I made the display by *cout* multiples of dash (-) and straight line (|) as the 3 x 3 board. Also, I used an array named *boardNumber* size = 9 to hold the board number from 1 to 9. For example, board number 1 is placed in row 1 column 1, board number 2 is in row 1 column 2 and so on. I learned to use set width function(*setw(int)*) to make my board align. I also practicing using array more.

* October 20th

In this progress I updated my hierarchy chart and explain briefly what functions I will use. Next, I added algorithms that players are able to mark their chosen number. Lets say, if player1 choose number 1 *boardNumber[0*] will be marked as ‘X’ or player2 choose number 5 *boardNumber[4]* will be marked as ‘O’. Also, I added function *bool check()* to validate winners. The objective of this game is to make three marks in a row. For example, board number 1,2 and 3 marked by player1 (‘X’). If the program found the winner, game will stop and display the winner. In this section, I learned and practice my logic skills.

* October 27th

This date, I improved my program by adding some validation for the inputs. Every input that is out of the range will display an error message. I also added draw function to check whether the game is draw. Next update, I will put class in my program.

* October 31st

I updated my draw function because I just realized that it does not works properly and I tried a lot of ways and searched on the Internet some methods that might help me to figure out my problem. I asked for Stavin’s help to find out what my problem is and with his guide I can manage it to works. I learned not to give up, eventually you will find a way to make it works.

* November 4th

I made a lot of changes to this program because I need to put class. I found it difficult to change my program from Procedural Programming to Object Oriented Programming because I just learned it only for a week. I divided the function that I made before into private and public members. Most of the functions are in the public members. On the private members, I put the array of the board number. I also added *menu,* *instruction* and *playAgain* function.

* November 6th

I finalized my coding. I re-read the coding and check all the validations for the last time.

* Overall, I learned that you have to follow your design and plan. Also, time management is crucial.

**3B. PROBLEMS**

The first problem that I found was choosing the right program I want to make. I have never learnt any programming language before, so it gave me a hard time to decide. Mr. Bagus also gave a deadline every week to update our project’s progress. This really pushed us to the limit. Because there are also other assignments we have to submit everyday, time management is one of my difficulties in the making of this program. Fortunately, I managed to upload the progress every time.

Logical errors also one of my difficulties, you can’t detect where I have gone wrong. Changing my program to object oriented program also is not easy. I had to run it many times to get it right. To overcome this problem, I read the book, searched methods, discussion about the problems and asked my colleagues and facilitators for advice.

**SECTION 4: SOURCE CODE**

* main.cpp

#include "TicTacToe.hpp"

#include <iostream>

#include <iomanip>

using namespace std;

bool menu();

int main()

{

bool start = false;

cout<<"\t\t\t\t\tWelcome to Tic-Tac-Toe Game!"<<endl<<endl;

start = menu();

while (start == true)

{

TicTacToe game;

int player = 1;

game.gamePlay(player);

game.playAgain();

}

* TicTacToe.hpp

#ifndef TicTacToe\_hpp

#define TicTacToe\_hpp

class TicTacToe

{

private:

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

public:

void boardGame();

bool check();

bool isBoardFilled();

void gamePlay(int playerPlay);

bool playAgain();

};

#endif /\* TicTacToe\_hpp \*/

* TicTacToe.cpp

#include "TicTacToe.hpp"

#include <iostream>

#include <iomanip>

using namespace std;

// game instruction

void instruction()

{

cout<<"What is Tic-Tac-Toe? \nIt is a game play on 3 x 3 board with two players. \nThe objective is to get three in a row."<<endl<<endl;

cout<<"How to play: \n";

cout<<"1. The first player is known for 'X' and second player is 'O'. \n\n";

cout<<"2. Players alternate placing Xs and Os on the game board until either oppent has three in a row or all nine squares are filled. \n\n";

cout<<"3. X always goes first, and in the event that no one has three in a row, the game is draw.\n\n";

}

// menu function

bool menu()

{

int input;

bool getInput = true;

bool start = false;

while (getInput == true)

{

cout<<"\t\tMain Menu\n\n";

cout<<"\t\t1. Play \n\n\t\t2. Instruction \n\n\t\t3. Quit \n"<<endl;

cout<<"Enter number 1, 2 or 3: ";

cin>>input;

cout<<endl;

if(input == 1)

{

getInput = false;

start = true;

}

else if(input==2)

{

char answer;

instruction();

cout<<"Back to Menu or quit (M/Q)? ";

cin>>answer;

if(answer == 'M' || answer == 'm')

start = true;

else if(answer == 'Q' || answer == 'q')

exit(0);

else

{

cout<<"ERROR: Invalid input. Please enter M or Q: ";

cin>>input;

}

}

else if (input == 3)

{

exit(0);

}

else

{

cout<<"ERROR: Invalid input. Please enter number 1, 2 or 3."<<endl;

}

}

return start;

}

// to replay the game

bool TicTacToe::playAgain()

{

bool start = true;

char input;

cout<<"Do you want to play again (Y/N)? ";

cin>>input;

if(input=='Y' || input == 'y')

{

start = menu();

}

else if(input=='N' || input == 'n')

{

cout<<"Thankyou for playing!"<<endl;

exit(0);

}

else

{

cout<<"ERROR: Invalid input. Please enter Y or N: ";

cin>>input;

cout<<endl;

}

return start;

}

// 3 x 3 board game

void TicTacToe::boardGame()

{

cout<<"\t\t\t\tPlayer1 (X) \t\t Player2 (O)"<<endl<<endl;

cout<<setw(41)<<" | | "<<endl;

cout<<setw(27)<<boardNumber[0]<<setw(3)<<"|"<<setw(3)<<boardNumber[1]<<setw(3)<<"|"<<setw(3)<<boardNumber[2]<<setw(3)<<endl;

cout<<setw(29)<<"\_\_\_\_\_"<<"|"<<"\_\_\_\_\_"<<"|"<<"\_\_\_\_\_"<<endl;

cout<<setw(41)<<" | | "<<endl;

cout<<setw(27)<<boardNumber[3]<<" "<<"|"<<setw(3)<<boardNumber[4]<<setw(3)<<"|"<<setw(3)<<boardNumber[5]<<setw(3)<<endl;

cout<<setw(29)<<"\_\_\_\_\_"<<"|"<<"\_\_\_\_\_"<<"|"<<"\_\_\_\_\_"<<endl;

cout<<setw(41)<<" | | "<<endl;

cout<<setw(27)<<boardNumber[6]<<setw(3)<<"|"<<setw(3)<<boardNumber[7]<<setw(3)<<"|"<<setw(3)<<boardNumber[8]<<setw(3)<<endl;

cout<<setw(41)<<" | | "<<endl;

}

// checking any three rows

bool TicTacToe::check()

{

//1,2,3

if (boardNumber[0] == boardNumber[1] && boardNumber[1] == boardNumber[2])

return true;

//1,5,9

else if (boardNumber[0] == boardNumber[4] && boardNumber[4] == boardNumber[8])

return true;

//1,4,7

else if (boardNumber[0] == boardNumber[3] && boardNumber[3] == boardNumber[6])

return true;

//2,5,8

else if (boardNumber[1] == boardNumber[4] && boardNumber[4] == boardNumber[7])

return true;

//3,5,7

else if(boardNumber[2] == boardNumber[4] && boardNumber[4] == boardNumber[6])

return true;

//3,6,9

else if(boardNumber[2] == boardNumber[5] && boardNumber [5] == boardNumber[8])

return true;

//4,5,6

else if(boardNumber[3] == boardNumber[4] && boardNumber[4] == boardNumber[5])

return true;

//7,8,9

else if(boardNumber[6] == boardNumber[7] && boardNumber[7] == boardNumber[8])

return true;

else

return false;

}

//checking draw game

bool TicTacToe::isBoardFilled()

{

bool filled = true;

for(int i = 0; i<9; i++)

{

if(!(boardNumber[i] == 'X' || boardNumber[i] == 'O'))

{

filled = false;

break;

}

}

return filled;

}

//game flow

void TicTacToe::gamePlay(int playerPlay)

{

//player1 starts first

playerPlay = 1;

//check wins

bool i = false;

//check draw

bool boardFilled = false;

while(i==false && boardFilled==false)

{

// Deciding the player

if (playerPlay % 2 == 0)

playerPlay = 2;

else if (playerPlay % 2 == 1)

playerPlay = 1;

boardGame();

int choice;

cout<<"Player"<<playerPlay<<" choose one number: ";

cin>> choice;

char mark = 'X';

// Deciding the marks, Player1 = X Player2 = O

if (playerPlay==1)

mark = 'X';

else if (playerPlay==2)

mark = 'O';

//mark the board

if (choice == 1 && boardNumber[0] == '1')

boardNumber[0] = mark;

else if (choice == 2 && boardNumber[1] == '2')

boardNumber[1] = mark;

else if (choice == 3 && boardNumber[2] == '3')

boardNumber[2] = mark;

else if (choice == 4 && boardNumber[3] == '4')

boardNumber[3] = mark;

else if (choice == 5 && boardNumber[4] == '5')

boardNumber[4] = mark;

else if (choice == 6 && boardNumber[5] == '6')

boardNumber[5] = mark;

else if (choice == 7 && boardNumber[6] == '7')

boardNumber[6] = mark;

else if (choice == 8 && boardNumber[7] == '8')

boardNumber[7] = mark;

else if (choice == 9 && boardNumber[8] == '9')

boardNumber[8] = mark;

else

{

cout<<"Invalid move "<<endl;

break;

}

//alternate players

playerPlay++;

i = check();

boardFilled = isBoardFilled();

}

if(i==true)

{

cout<<"Player"<<--playerPlay<<" wins!"<<endl;

}

else if(boardFilled==true)

{

cout << "Draw game!"<<endl;

}

}