Coursework 2

Name:

Koay Xian Cong

Student ID:

20418760

Module:

EEEE1042 Software Engineering

Lecturer:

Dr Chan Kheong Sann

Topic:

Tic-Tac-Toe

In Question 3, we are going to let computer play against itself and human is the only observer. I have created two types of AI in Question 2:

1. Smart AI
2. Random AI

to gather the statistics of numbers of wins, draws, and losses for between Smart AI and Random AI. The following are how I did the experiment:

<u>Generating Random Numbers</u>

```
int randomNum() { //function to generate random number
      int random = 1 + (9.0*rand())/(RAND_MAX+1.0); //generate with MSB formula
      return random;
  }
```

Firstly, I created a random number generator function which creates random number between 1-9. I used the following equation:

```
1 + (9.0*rand())/(RAND_MAX+1.0)
```

which is the MSB formula to generate the random numbers as the it is proven in previous practical that it will generate an equal number of frequencies of random values. Hence, the biasness of function generating higher frequency of same random numbers over time can be reduced.

<u>Random AI Input</u>

```
void computerRand(char XO, char *grid, int *store) { //function to generate randomAI
  int nonLegal=1; //check if the move is legal and satisfy or not
    while(nonLegal){ //infinity loop to generate random values if moves are not legal
      int random = randomNum(); //store it as random
      if (store[random]==random){ //check if it is legal moves, if there are things stored
in the temperory store array, it will loop again and get new random values
        nonLegal=1; //generate new random values
      }
      else{ //if it is legal moves and the temporary store array is default value(10), it
will store 'X' into grid and store value generated into temporary store array
        grid[random]=XO; //store into grid if it's legal moves
        store[random]=random; //replace default value with random value generated
        nonLegal=0; //break the infinity loop
      }
    }
}
```

Next, I created a function to generate the random AI. In this function, I will first check if the value generated is legal moves or not by checking the store array whether the value matches with the random value generated. If it matches, it means that the space is being used and the computer will loop into non legal moves again and generate new random number. If the move is legal, it will store the value into the store array and print ‘X’ or ‘O’ into grid array depends on which computers’ turn. Finally, it will break the condition loop to prepare for next function to be called.

<u>Generating Smart AI Algorithm</u>

```
int AI(char *grid, char XO) { //function computer AI to check whether to attack or block
    if ((grid[1] == grid[2]) && grid[3]==' ' && grid[1]==XO) return 3;
    else if ((grid[4] == grid[5]) && grid[6]==' ' && grid[4]==XO) return 6;
    else if ((grid[7] == grid[8]) && grid[9]==' ' && grid[7]==XO) return 9;

    else if ((grid[2] == grid[3]) && grid[1]==' ' && grid[2]==XO) return 1;
    else if ((grid[5] == grid[6]) && grid[4]==' ' && grid[5]==XO) return 4;
    else if ((grid[8] == grid[9]) && grid[7]==' ' && grid[8]==XO) return 7;

    else if ((grid[1] == grid[3]) && grid[2]==' ' && grid[1]==XO) return 2;
    else if ((grid[4] == grid[6]) && grid[5]==' ' && grid[4]==XO) return 5;
    else if ((grid[7] == grid[9]) && grid[8]==' ' && grid[7]==XO) return 8;

    else if ((grid[1] == grid[4]) && grid[7]==' ' && grid[1]==XO) return 7;
    else if ((grid[2] == grid[5]) && grid[8]==' ' && grid[2]==XO) return 8;
    else if ((grid[3] == grid[6]) && grid[9]==' ' && grid[3]==XO) return 9;

    else if ((grid[4] == grid[7]) && grid[1]==' ' && grid[4]==XO) return 1;
    else if ((grid[5] == grid[8]) && grid[2]==' ' && grid[5]==XO) return 2;
    else if ((grid[6] == grid[9]) && grid[3]==' ' && grid[6]==XO) return 3;

    else if ((grid[1] == grid[7]) && grid[4]==' ' && grid[1]==XO) return 4;
    else if ((grid[2] == grid[8]) && grid[5]==' ' && grid[2]==XO) return 5;
    else if ((grid[3] == grid[9]) && grid[6]==' ' && grid[3]==XO) return 6;

    else if ((grid[1] == grid[5]) && grid[9]==' ' && grid[1]==XO) return 9;
    else if ((grid[5] == grid[9]) && grid[1]==' ' && grid[5]==XO) return 1;
    else if ((grid[1] == grid[9]) && grid[5]==' ' && grid[1]==XO) return 5;

    else if ((grid[3] == grid[5]) && grid[7]==' ' && grid[3]==XO) return 7;
    else if ((grid[5] == grid[7]) && grid[3]==' ' && grid[5]==XO) return 3;
    else if ((grid[3] == grid[7]) && grid[5]==' ' && grid[3]==XO) return 5;
    else return -2;
}
```

Subsequently, I generated a function which will be the algorithm of my AI.

1. If the computer can immediately win on its current move, it will play the winning move and ends the game.
2. If the opponent is about to win on the next move, it can block the opponents’ winning move.
3. If neither above 2 scenarios occurs, the computer falls back to the random AI.

This is known as reactionary play. Reactionary players will block their opponents three in a row or take any three in a row that they can. Otherwise, they will choose random moves.[1] In my code, the computer will return -2 if there isn't any condition satisfied. If the computer finds the moves satisfied and if the move is legal by checking if there are empty spaces in the grid, it will return the moves' value.

Smart AI Input

```c
int computerAI(char XO, char *grid, int *store) { //function to call SmartAI
  int nonLegal = 1; //check if the move is legal and satisfy or not
  char attackBlock[2]; //array to check if computer need to attack or block
  if(XO == 'X') { //check if computer is 'X' or 'O'
    attackBlock[0]='X'; //attack[0] is used for attack
    attackBlock[1]='O'; //attack[1] is used for block
  }
  else {
    attackBlock[0]='O';
    attackBlock[1]='X';
  }
  int attack=AI(grid, attackBlock[0]); //Computer will loop through the AI algorithm to check if it can win or not and return values as attack
  int block=AI(grid, attackBlock[1]); //it will loop through the AI algorithm to check if it can block or not
  if (attack!=-2) { //if there is condition satisfied, the computer will attack and win the game first when it returns value not equals to -2
    grid[attack]=XO; //place 'X' or 'O' depends on computers to attack and win the game
    store[attack]=attack; //store value into temperory store function to match with player input hence it will only play without overwritting the other computer
  }
  else if (block!=-2) { //if there is no winning condition first, the computer will look for blocking condition and block if it satisfied the condtion and return value which is not equals to -2
    grid[block]=XO; //place 'X' or 'O' depends on computers to block to prevent player from winning
    store[block]=block; //store value into temperory store function to match with player input hence it will only play without overwritting the player
  }
  else{ //if value returns is -2, it falls back to randomAI and chooses legal move.
    while(nonLegal) { //infinity loop to generate random values if moves are not legal
      int random=randomNum(); //store it as random
      if (store[random]==random) { //check if it is legal moves, if there are things stored in the temperory store array, it will loop again and get new random values
        nonLegal=1; //generate new random values
      }
      else{ //if it is legal moves and the temporary store array is default value(10), it will store 'X' into grid and store value generated into temprorary store array
        grid[random]=XO; //store into grid if it's legal moves
        store[random]=random; //replace default value with random value generated
        nonLegal=0; //break the infinity loop
      }
    }
  }
}
```

Then, I created a function to generate the smartAI. In this function, the computer will first parse 'X' and 'O' characters to check when to attack and block in the attackBlock array. attackBlock[0] is used to play the attack move while attackBlock[1] is used to play the block move for both computers. The computer will check in the previous AI algorithm function to see if it should attack or block. The AI algorithm which returns the value will be used to insert 'X' or 'O' depending on which computers in the grid array and store the value in the store array so that it can check whether the moves is legal or not for next iteration. The computer will perform attack moves first if it checks that it can win the game followed by blocking moves if there isn't attacking moves. If the AI algorithm returns -2 value, that means there are neither attack nor blocking conditions fulfilled. The computer will then generate random values and check if the value generated is legal moves or not by checking the store array whether it matches with the random value generated. Once again, if it matches, it means that the space is being used and the computer will loop into non legal moves again and generate new random number. If the move is legal, it will store the value into the store array with the value generated and print 'X' or 'O' into grid array depending on which computers' turn. Finally, it will break the condition loop to prepare for next function to be called.

Integrating Both Smart AI and Random AI

```
void typeOfPlayers(int *condition, int types){ //function to loop over different types of computers(i.e. smart and random)
  int i=1;  //variable to iterate over the loop.
  char XO; //switch between XO for each player turn
  int store[]={10,10,10,10,10,10,10,10,10,10}; //array use to check user input
  char grid[gridSize]; //array use to store tic tac toe board

  setEmpty(grid); //clear the tic-tac-toe board with white space

  int computer=1; //set which computer plays first

  while(i<gridSize){
    if (computer%2 == 0){ //loop between computer 1 and 2
      switch(types) { //loop from main from 1 to 3
        case 1: //when go through the first iteration in the loop, smartAI is chosen
          computerAI('X',grid,store);
          break;
        case 2: //when go through the second iteration in the loop, smartAI is chosen
          computerAI('X',grid,store);
          break;
        case 3: //when go through the third iteration in the loop, randomAI is chosen
          computerRand('X',grid,store);
          break;
      }
      XO='X'; //computer 1-'X'
    }
    else{ //loop between computer 1 and 2
      switch(types) {
        case 1: //when go through the first iteration in the loop, smartAI is chosen
```

```
          computerAI('O',grid,store);
          break;
        case 2: //when go through the second iteration in the loop, randomAI is chosen
          computerRand('O',grid,store);
          break;
        case 3: //when go through the third iteration in the loop, randomAI is chosen
          computerRand('O',grid,store);
          break;
      }
      XO='O'; //computer 2 - 'O'
    }

      //Will exit the loop if checkWin function return 1.
      if (checkWin(grid)){ //go through checkWin function to see if any computer has won
        i=20;
        if (XO == 'X') condition[2]++; //Player x wins the game and stored in condition[2]
        else condition[0]++; //Player o wins the game and stored in condition[0]
      }
      else{ //if checkWin function is false, continue to loop
        computer++;
        i++;
      }
  }
  if (i != 20){ //Draw if no one win the game
    condition[1]++; //Store tie games in condition[1]
  }
}
```

Next, I created a function to loop between the 3 situations as per stated earlier which are:

Smart AI vs Smart AI – Type 1

Smart AI vs Random AI – Type 2

Random AI vs Random AI – Type 3

Computer is being defined 1 at first hence computer ‘O’ will start the game. The computer will check the winning condition and print into condition[1] if computer ‘O’ wins, condition[2] if draw and condition[3] if computer ‘X’ wins.

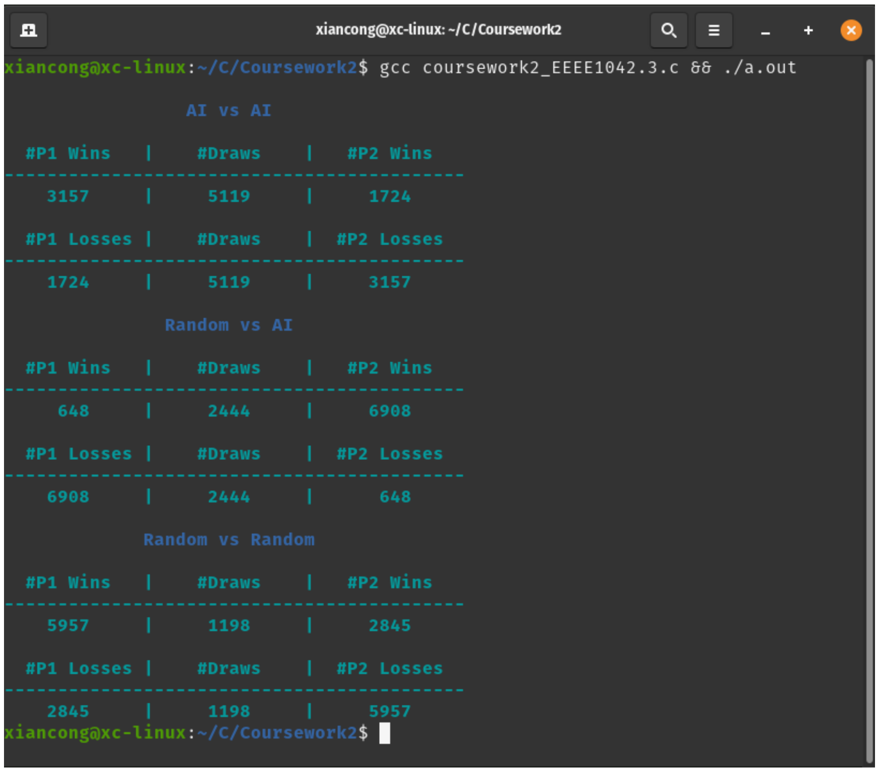
```
int main(){
  int types; //variable to store smart vs smart, smart vs random and random vs random
  for(int i=0;i<3;i++){
    int condition[]={0,0,0}; //store the number of times for computer wins, draw or losses
    srand(clock()); //get random seed from clock function hence its randomized everytime
    switch(i) {
      case 0: //smart vs smart
        types=1;
        break;
      case 1: //smart vs random
        types=2;
        break;
      case 2: //random vs random
        types=3;
        break;
    }
    for(int j=0;j<10000;j++) {
      typeOfPlayers(condition,types); //iterate over 10000 times to get better accuracy of results
    }
    result(types, condition); //calling result function to print result
  }
}
```

In the main function, the computer will loop times to generate 3 different types of result. Random seed is created with:

```
srand(clock());
```

to generate different types of seed everytime as clock() function take current time in milliseconds and uses that as a start seed. If clock() function is not being used, the random values will be consistent and the same in every iterations. Finally, the computer will loop through the 3 different types of situations 10000 times to generate a result with better accuracy to show that Smart AI performs better than Random AI.

The results are then printed and shown in Figure 1.



```
xiancong@xc-linux:~/C/Coursework2$ gcc coursework2_EEEE1042.3.c && ./a.out

                AI vs AI

  #P1 Wins    |     #Draws     |    #P2 Wins
-------------------------------------------
    3157      |      5119      |      1724

  #P1 Losses  |     #Draws     |   #P2 Losses
-------------------------------------------
    1724      |      5119      |      3157

              Random vs AI

  #P1 Wins    |     #Draws     |    #P2 Wins
-------------------------------------------
     648      |      2444      |      6908

  #P1 Losses  |     #Draws     |   #P2 Losses
-------------------------------------------
    6908      |      2444      |       648

            Random vs Random

  #P1 Wins    |     #Draws     |    #P2 Wins
-------------------------------------------
    5957      |      1198      |      2845

  #P1 Losses  |     #Draws     |   #P2 Losses
-------------------------------------------
    2845      |      1198      |      5957
xiancong@xc-linux:~/C/Coursework2$
```

Figure 1: Results of Smart AI vs Smart AI (Top), Random AI vs Smart AI (Centre), Random AI vs Random AI(Bottom)

<u>**Analysis of Smart AI vs Smart AI**</u>

Smart AI vs Smart AI has higher percentage of drawing the game as both AI can block and attack depending on the conditions. Based on the result, first Smart AI which starts the game has higher percentage of winning the game because it will have more moves by the end of the game.

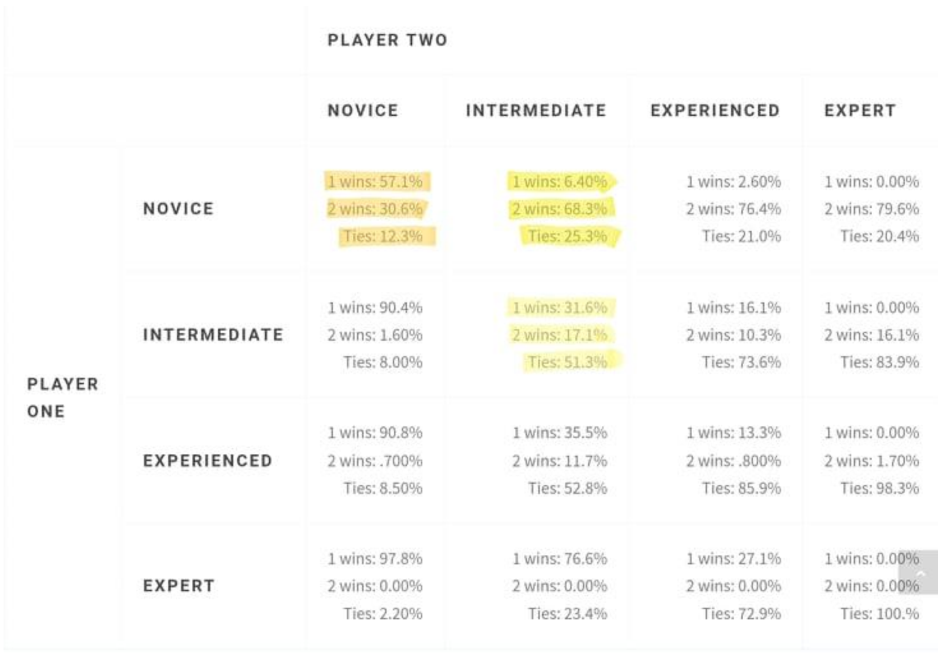
<u>Analysis of Smart AI vs Random AI</u>

Smart AI has way higher percentage of winning in the Random AI vs Smart AI as #P2 is the Smart AI. This is because the Smart AI can attack and ends the game if there is condition that matches in the algorithm. Smart AI can also block the Random AI winning move.

<u>Analysis of Random AI vs Random AI</u>

Random AI which makes the first move will have won about twice as often as the player that goes second which is proven in the result from Figure 1. This suggest that first player which makes the move has advantage against second player.

My result is found to be closed to research finding about Tic Tac Toe winning strategies as show in Figure 2:



		PLAYER TWO			
		NOVICE	INTERMEDIATE	EXPERIENCED	EXPERT
PLAYER ONE	NOVICE	1 wins: 57.1% 2 wins: 30.6% Ties: 12.3%	1 wins: 6.40% 2 wins: 68.3% Ties: 25.3%	1 wins: 2.60% 2 wins: 76.4% Ties: 21.0%	1 wins: 0.00% 2 wins: 79.6% Ties: 20.4%
	INTERMEDIATE	1 wins: 90.4% 2 wins: 1.60% Ties: 8.00%	1 wins: 31.6% 2 wins: 17.1% Ties: 51.3%	1 wins: 16.1% 2 wins: 10.3% Ties: 73.6%	1 wins: 0.00% 2 wins: 16.1% Ties: 83.9%
	EXPERIENCED	1 wins: 90.8% 2 wins: .700% Ties: 8.50%	1 wins: 35.5% 2 wins: 11.7% Ties: 52.8%	1 wins: 13.3% 2 wins: .800% Ties: 85.9%	1 wins: 0.00% 2 wins: 1.70% Ties: 98.3%
	EXPERT	1 wins: 97.8% 2 wins: 0.00% Ties: 2.20%	1 wins: 76.6% 2 wins: 0.00% Ties: 23.4%	1 wins: 27.1% 2 wins: 0.00% Ties: 72.9%	1 wins: 0.00% 2 wins: 0.00% Ties: 100.%

Figure 2: Research Findings by Stephen Ostermiller, from the Tic Tac Toe Strategy

Novice player in the research finding is the same as my Random AI which values were inserted randomly while the intermediate player is the same as my reactionary Smart AI which blocks and attacks depending on conditions. Otherwise, the computer will play random moves. The research finding looped through 1000 iterations and get the percentage as shown in Figure 2.

In conclusion, the result from my smart AI is refined but not flawless, this is because the smart AI is only able to make moves based on last move and decide whether to block or attack. There are no strategies involved such as where are the better starting placements nor predicting the next move of another player or computer. However, the smart AI is smart enough to end the game and block the opponents when the condition matches.

Reference

1. *Tic-tac-toe strategy*. Stephen Ostermiller. (2015, March 6). Retrieved December 18, 2021, from https://blog.ostermiller.org/tic-tac-toe-strategy/