Name : Mihir Thakkar
Roll no. 59
SRN : 201901267

Artificial Intelligence Ass1- Implementation of Non AI techniques

1a. TicTacToe using Magic Square :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
char board[5][5] = {0};
int
arr[]={1,2,3,4,5,6,7,8,9};
int
mag[]={8,3,4,1,5,9,6,7,2
};
int arr2[]={-1};
int
pla[5],comp[5];
int m;
int check()
{ int
i=0;
 while(i<9)
 {
 if(arr2[i] == 1 && arr2[i+1] == 1 && arr2[i+2] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!"); return 1;
 }
 if(arr2[i] == 2 && arr2[i+1] == 2 && arr2[i+2] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!"); return 1;
 }
 i =
i+3; }
 i=0;
while(i<3)
 {
 if(arr2[i] == 1 && arr2[i+3] == 1 && arr2[i+6] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+3] == 2 && arr2[i+6] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!");
 return 1;
 }
i++; }
 i=0;
 if(arr2[i] == 1 && arr2[i+4] == 1 && arr2[i+8] == 1)
```

```
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+4] == 2 && arr2[i+8] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!"); return 1;
 }
 i=2;
 if(arr2[i] == 1 && arr2[i+2] == 1 && arr2[i+4] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+2] == 2 && arr2[i+4] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!"); return 1;
 }
 return 0;
}
void dis_board()
{
 int i,j;
 for(i=0;i<5;i++)
 {
 for(j=0;j<5;j++)
 {
 printf("\t%c",board[i][j]);
 }
 printf("\n");
 }
}
int main()
{
 printf("\t\t\t\t\t\tTIC TAC TOE!!\n\n\n\n"); int
i,j,p,c=0,mc1=0,mc2=0,a; int move=1;
 printf("\n\n1. Player plays first\n2. Computer plays first\nYour Choice : "); scanf("%d",&i); if(i==1)
p=1; else p=2;
for(i=1;i<5;i++)
 {
 for(j=0;j<5;j++)
board[i][j] = '-';
 i++;
 }
 for(j=1;j<5;j++)
 {
 for(i=0;i<5;i++)
board[i][j] = '|';
 j++;
 }
 dis_board();
m=0;
while(m<9)
```

```
 {
 j=0;
 printf("\n\nAvailable Moves : \n\n");
 for(i=0;i<9;i++)
 {
 printf("\t%c",arr[i] + 48);
 j++;
if(j==3)
 {
printf("\n");
j=0;
 }
 }
 if(p==1)
 {
 printf("\n\t\t\tPlayer 1(X) : \n\nChoose your move index : "); scanf("%d",&move);
printf("\n\n");
arr[move-1] = -3;
arr2[move-1] = 1;
pla[mc2] = mag[move-1];
 mc2++;
switch(move)
 { case 1:
board[0][0] = 'X';
 break;
case 2:
board[0][2] = 'X';
 break;
case 3:
board[0][4] = 'X';
 break;
case 4:
board[2][0] = 'X';
 break;
case 5:
board[2][2] = 'X';
 break;
case 6:
board[2][4] = 'X';
 break;
case 7:
board[4][0] = 'X';
 break;
case 8:
board[4][2] = 'X';
 break;
case 9:
board[4][4] = 'X';
break;
 }

dis_board();
c = check();
```

```
if(c==1)
return 0;
p++;
m++;
 }
else
 {
 printf("\n\t\t\tPlayer COMP(O) : \n\nChoose your move index : "); if(mc1==0)
 {
 move = 5;
if(arr[move-1] == -3)
 goto jump2;
 }
 if(mc1==1)
 {
jump2:
 move = 1;
 while(arr[move-1] == -3 && move<9)
 move = (move*2) + 1;
 }
 if(mc1>=2)
 {
 a = 15 - comp[mc1-1] -
comp[mc1-2]; if(a<9)
{
 i=0;
 while(mag[i] !=
a) i++;
 move = i+1;
 }
else
goto jump1;
 if(arr[move-1] == -3)
 {
 a = 15 - pla[mc2-1] -
pla[mc2-2]; if(a<9)
{
 i=0;
 while(mag[i] != a)
 i++;
move = i+1;
 }
else
goto jump1;
 }
 jump1:
 if(arr[move-1] == -3 || move>=9)
 {
 while(arr[move-1] == -3)
 move = rand() %9 + 1;
 }
 }
```

```
printf("%d\n\n",move);
arr[move-1] = -3;
arr2[move-1] = 2;
comp[mc1] = mag[move-1];
mc1++; switch(move)
 { case 1:
board[0][0] = 'O';
break; case 2:
board[0][2] = 'O';
break; case 3:
board[0][4] = 'O';
break; case 4:
board[2][0] = 'O';
break; case 5:
board[2][2] = 'O';
break; case 6:
board[2][4] = 'O';
break; case 7:
board[4][0] = 'O';
break; case 8:
board[4][2] = 'O';
break;
 case 9:
board[4][4] = 'O';
 break;
 }

dis_board();
c = check();
if(c==1)
return 0;
p--; m++;
 }
 }
 if(m==9)
 printf("\n\n\t\t\t\t\tTHAT'S A DRAW!!");
 return 0;
}
```

Output : -

```
1. Player plays first
2. Computer plays first
Your Choice : 1
                |               |
        -       |       -       |       -
                |               |
        -       |       -       |       -
                |               |


Available Moves :

        1       2       3
        4       5       6
        7       8       9

                        Player 1(X) :

Choose your move index : 1


        X       |               |
        -       |       -       |       -
                |               |
        -       |       -       |       -
                |               |


Available Moves :

        -       2       3
        4       5       6
        7       8       9
```

```
                        Player COMP(O) :

Choose your move index : 5

        X       |               |
        -       |       -       |       -
                |       O       |
        -       |       -       |       -
                |               |


Available Moves :

        -       2       3
        4       -       6
        7       8       9

                        Player 1(X) :

Choose your move index : 3


        X       |               |       X
        -       |       -       |       -
                |       O       |
        -       |       -       |       -
                |               |


Available Moves :

        -       2       -
        4       -       6
        7       8       9
```

```
                        Player COMP(O) :

Choose your move index : 7


        X       |               |       X
        -       |       -       |       -
                |       O       |
        -       |       -       |       -
        O       |               |


Available Moves :

        -       2       -
        4       -       6
        -       8       9

                        Player 1(X) :

Choose your move index : 2


        X       |       X       |       X
        -       |       -       |       -
                |       O       |
        -       |       -       |       -
        O       |               |


                                        Player 1 WINS!!
Process returned 0 (0x0)   execution time : 13.458 s
Press any key to continue.
```

1b. Implementation of Magic Square :

```
#include<stdio.h>
int main()
{
 printf("\t\t\t\t\t\t\tMAGIC SQUARE!!\n\n\n\n"); int n;
 printf("\nEnter an Odd Number : ");
scanf("%d",&n);
int arr[n][n];
 memset(arr, 0, sizeof(arr[n][n]) * n * n);
 int i=0,j,val=1;
int
temp1,temp2;
j = (n-1)/2;
arr[i][j] = val;
val++;
while(val<=(n*n)
)
 {
 temp1 = i;
temp2 = j;
 i--; if(i
== -1) i =
n-1; j++;
if(j == n) j
= 0;
if(arr[i][j] == 0)
arr[i][j] = val;
else
 {
 i =
temp1+1; j
= temp2;
 arr[i][j] = val;
 }
 val++;
 }
 printf("\n\n");
 for(i=0;i<n;i++)
 {
 for(j=0;j<n;j++)
 printf("\t%d",arr[i][j]);
printf("\n");
 }
 return 0;
}
```

Output:-

```
Enter an Odd Number : 3

        8       1       6
        3       5       7
        4       9       2

Process returned 0 (0x0)   execution time : 2.938 s
Press any key to continue.
```

1c. N-Queens problem :

```
#include<stdio.h>
int arr[10][10] = {0};
int q = 1,n;
void place()
{ int
i,j,k,l,temp;
i=0;
 j = 0;
 while(q != n+1)
 {
 if(arr[i][j] == 0)
 {
arr[i][j] = q;
k = j + 1;
if(k == n)
k = 0;
while(k != j)
 {
 if(arr[i][k] == 0)
arr[i][k] = -q;
k++;
if(k == n)
    k = 0;
 }
k = i + 1;
if(k == n)
k = 0;
while(k != i)
 {
 if(arr[k][j] == 0)
arr[k][j] = -q;
k++;
```

```
if(k == n)
    k = 0;
 }
k = i + 1;
l = j + 1;
if(k == n &&
k>=j)
 {
k=k-l;
l=0; }
else
 {
 if(l == n && l>k)
 {
l=l-k;
k=0;
}
 }
 while(k != i)
 {
if(arr[k][l] == 0)
arr[k][l] = -q;
k++;
l++;
if(k == n && k>=j)
 {
k=k-l;
l=0;
}
else
 {
 if(l == n && l>k)
 {
l=l-k;
k=0;
}
 }
}
k = i + 1;
l = j - 1;
if(l == -1)
{
l=l+k;
k=0;
}
else {
if(k == n)
 {
 temp = l;
l=k-1;
k=k-l+temp;
 }
 }
```

```
 while(k != i)
 {
if(arr[k][l] == 0)
arr[k][l] = -q;
k++;
l--;
if(l == -1)
{
l=l+k;
k=0;
}
else {
if(k == n)
{
    temp = l; l=k-1;
 k=k-l+temp;
 }
 }
}
j = q;
i = 0;
q++;
} else
{
 i++;
if(i == n && j == q-1)
 {
 for(k=0;k<n;k++)
 {
 for(l=0;l<n;l++)
 {
if(arr[k][l] == -j)
arr[k][l] = 0;
if(arr[k][l] == j)
arr[k][l] = -1;
 }
 }
q--;
i=0;
j=q-1;
 }
 }
 }
}
int main()
{
 printf("\t\t\t\t\t\t\tN QUEENS PROBLEM!!\n\n\n\n"); int i,j;
 printf("\nEnter the Number of Queens : "); scanf("%d",&n);
printf("\n\n");
char dis[n][n];
place(); for(i=0;i<n;i++)
 {
 for(j=0;j<n;j++)
```

```
 {
if(arr[i][j]>0)
dis[i][j] = 'Q';
else
dis[i][j] = '-';
 }
 }
 for(i=0;i<n;i++)
 {
 for(j=0;j<n;j++)
 {
 printf("\t\t%c",dis[i][j]);
 }
 printf("\n");
 }
 return 0;
}
```

Output:-

```
                                                N QUEENS PROBLEM!!



Enter the Number of Queens : 5

                Q               -               -               -               -
                -               -               -               Q               -
                -               Q               -               -               -
                -               -               -               -               Q
                -               -               Q               -               -

Process returned 0 (0x0)   execution time : 1.420 s
Press any key to continue.
```

1d. TicTacToe using MinMax algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<iostream>

using namespace std;
int
arr[]={1,2,3,4,5,6,7,8,9};
int arr2[9]={-1};
int m;
struct Move
{
int row, col;
};
char player = 'x', opponent = 'o';
int check()
{ int i=0;
while(i<9)
 {
 if(arr2[i] == 1 && arr2[i+1] == 1 && arr2[i+2] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+1] == 2 && arr2[i+2] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!");
 return 1;
 }
i = i+3;
}
 i=0;
while(i<3)
 {
 if(arr2[i] == 1 && arr2[i+3] == 1 && arr2[i+6] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+3] == 2 && arr2[i+6] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!");
 return 1;
 }
i++; }
 i=0;
 if(arr2[i] == 1 && arr2[i+4] == 1 && arr2[i+8] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+4] == 2 && arr2[i+8] == 2)
```

```
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!"); return 1;
 }
 i=2;
 if(arr2[i] == 1 && arr2[i+2] == 1 && arr2[i+4] == 1)
 {
 printf("\n\n\t\t\t\t\tPlayer 1 WINS!!");
 return 1;
 }
 if(arr2[i] == 2 && arr2[i+2] == 2 && arr2[i+4] == 2)
 {
 printf("\n\n\t\t\t\t\tCOMPUTER WINS!!"); return 1;
 }
 return 0;
}
bool isMovesLeft(char board[5][5])
{
for (int i = 0; i<5; i++)
for (int j = 0; j<5;
j++)
if (board[i][j]==' ')
return true;
return false;
}
int evaluate(char b[5][5])
{
for (int row = 0; row<5; row++)
{
if (b[row][0]==b[row][2] && b[row][2]==b[row][4])
{
if (b[row][0]==player)
return +10;
else if (b[row][0]==opponent)
return -10;
}
}
for (int col = 0; col<5; col++)
{
if (b[0][col]==b[2][col] && b[2][col]==b[4][col])
{
if (b[0][col]==player)
return +10;
else if (b[0][col]==opponent)
return -10;
}
}
if (b[0][0]==b[2][2] && b[2][2]==b[4][4])
{
 if (b[0][0]==player) return
+10;
else if (b[0][0]==opponent)
return -10;
}
```

```
if (b[0][4]==b[2][2] && b[2][2]==b[4][0])
{
if (b[0][4]==player)
return +10;
else if (b[0][4]==opponent)
return -10;
}
return 0;
}
int minimax(char board[5][5], int depth, bool isMax)
{
int score =
evaluate(board); if
(score == 10)
return score;
if (score == -10)
return score;
if (isMovesLeft(board)==false)
return 0;
if (isMax)
{
int best = -1000;
for (int i = 0; i<5; i++)
{
for (int j = 0; j<5; j++)
{
if (board[i][j]==' ')
{
board[i][j] = player;
best = max( best,minimax(board, depth+1, !isMax)
); board[i][j] = ' ';
}
}
}
return
best; }
else {
int best = 1000;
for (int i = 0; i<5; i++)
{
for (int j = 0; j<5; j++)
{
if (board[i][j]==' ')
{
board[i][j] = opponent;
best = min(best,minimax(board, depth+1, !isMax));
board[i][j] = ' ';
}
}
}
return best;
}
}
```

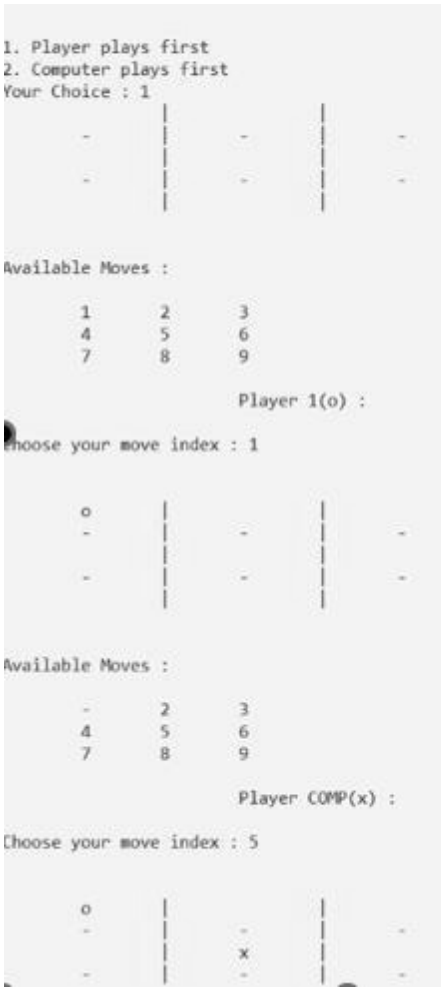
```
Move findBestMove(char board[5][5])
{
int bestVal = -1000;
Move bestMove;
bestMove.row = -1;
bestMove.col = -1;
for (int i = 0; i<5; i++)
{
for (int j = 0; j<5; j++)
{
 if (board[i][j]==' ')
{
board[i][j] = player;
int moveVal = minimax(board, 0, false);
board[i][j] = ' ';
if (moveVal > bestVal)
{
bestMove.row = i;
bestMove.col = j;
bestVal = moveVal;
}
}
}
}
//printf("\n\nThe value of the best Move is :%d\n",bestVal); return bestMove;
}
void dis_board(char board[5][5])
{
 int i,j;
 for(i=0;i<5;i++)
 {
 for(j=0;j<5;j++)
 {
 printf("\t%c",board[i][j]);
 }
 printf("\n");
 }
}
int main()
{
char board[5][5] =
{
{' ','|',' ','|',' '},
{'-','|','-','|','-'},
{' ','|',' ','|',' '},
{'-','|','-','|','-'},
{' ','|',' ','|',' '}
};
printf("\t\t\t\t\t\t\tTIC TAC TOE!!\n\n\n\n");
int i,j,p,c=0;
int mover=1;
 printf("\n\n1. Player plays first\n2. Computer plays first\nYour Choice : ");
scanf("%d",&i);
```

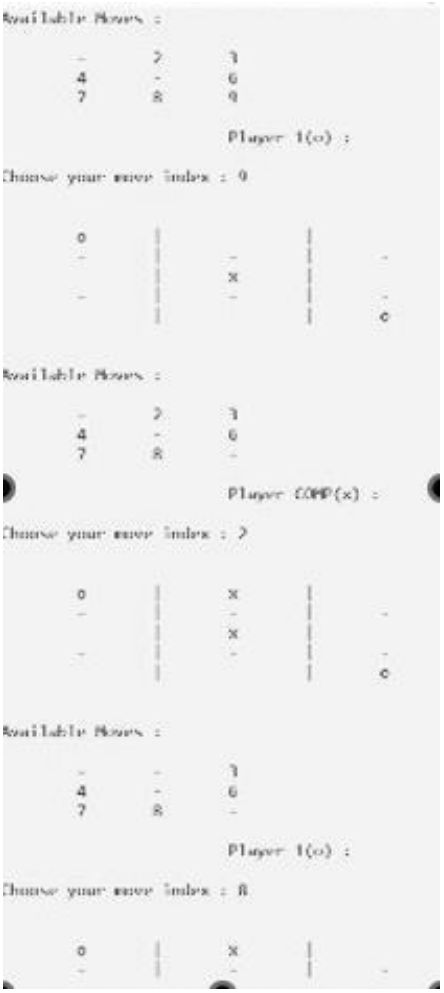
```
if(i==1)
    p=1;
else
 p=2;
 dis_board(board);
 m=0;
while(m<9)
 {
 j=0;
 printf("\n\nAvailable Moves : \n\n");
 for(i=0;i<9;i++)
 {
 printf("\t%c",arr[i] +
48); j++;
if(j==3)
 {
 printf("\n");
j=0;
 }
 }
 if(p==1)
 {
 printf("\n\t\t\tPlayer 1(o) : \n\nChoose your move index : ");
 scanf("%d",&mover);
printf("\n\n");
arr[mover-1] = -3;
arr2[mover-1] = 1;
 switch(mover)
 { case 1:
board[0][0] = 'o';
 break;
case 2:
board[0][2] = 'o';
 break;
case 3:
board[0][4] = 'o';
 break;
case 4:
board[2][0] = 'o';
 break;
case 5:
board[2][2] = 'o';
 break;
case 6:
board[2][4] = 'o';
 break;
case 7:
board[4][0] = 'o';
 break;
case 8:
board[4][2] = 'o';
 break;
case 9:
```

```
board[4][4] = 'o';
 break;
 }

dis_board(board);
c = check();
if(c==1)
    return 0;
    p++;
 m++;
 }
else
 {
 printf("\n\t\t\tPlayer COMP(x) : \n\nChoose your move index : ");
 Move bestMove = findBestMove(board);
board[bestMove.row][bestMove.col] = 'x';
 if(bestMove.row == 0)
 {
 if(bestMove.col == 0)
mover = 1;
if(bestMove.col == 2)
mover = 2;
if(bestMove.col == 4)
mover = 3;
 }
 if(bestMove.row == 2)
 {
 if(bestMove.col == 0)
mover = 4;
if(bestMove.col == 2)
mover = 5;
if(bestMove.col == 4)
mover = 6;
 }
 if(bestMove.row == 4)
 {
 if(bestMove.col == 0)
mover = 7;
if(bestMove.col == 2)
mover = 8;
if(bestMove.col == 4)
mover = 9;
 }

printf("%d\n\n\n",mover);
arr[mover-1] = -3;
arr2[mover-1] = 2;
dis_board(board);
c = check();
if(c==1)
    return 0;
    p--;
m++;
```

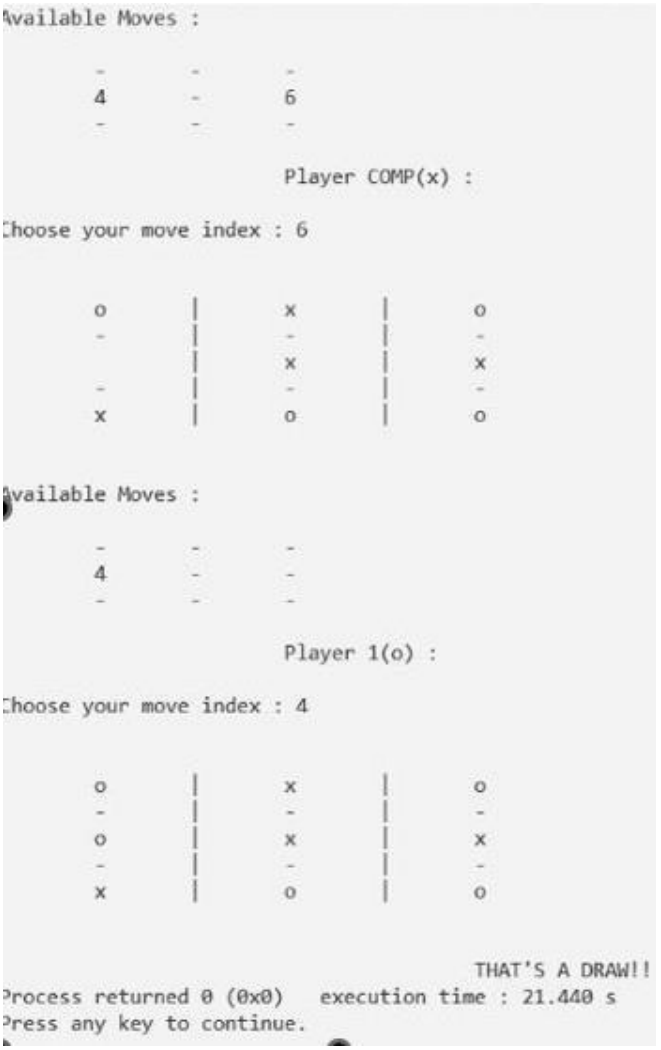
```
 }
 }
 if(m==9)
 printf("\n\n\t\t\t\t\tTHAT'S A DRAW!!");
 return 0;
}
```

Output:-





```
                    Player 1(o) :

Choose your move index : 8


      o     |     x     |
      -     |     -     |     -
            |     x     |
      -     |     -     |     -
            |     o     |     o

Available Moves :

      -     -     3
      4     -     6
      7     -     -

                    Player COMP(x) :

Choose your move index : 7


      o     |     x     |
      -     |     -     |     -
            |     x     |
      -     |     -     |     -
      x     |     o     |     o

Available Moves :

      -     -     3
      4     -     6
      -     -     -

                    Player 1(o) :

Choose your move index : 3


      o     |     x     |     o
      -     |     -     |     -
            |     x     |
      -     |     -     |     -
      x     |     o     |     o

Available Moves :
```



```
Available Moves :

      -     -     -
      4     -     6
      -     -     -

                    Player COMP(x) :

Choose your move index : 6


      o     |     x     |     o
      -     |     -     |     -
            |     x     |     x
      -     |     -     |     -
      x     |     o     |     o


Available Moves :

      -     -     -
      4     -     -
      -     -     -

                    Player 1(o) :

Choose your move index : 4


      o     |     x     |     o
      -     |     -     |     -
      o     |     x     |     x
      -     |     -     |     -
      x     |     o     |     o


                                    THAT'S A DRAW!!
Process returned 0 (0x0)   execution time : 21.440 s
Press any key to continue.
```