(TIC-TAC-TOE)

DESCRIPTION-

I selected a tic tac toe game.

Structure of code-

User has to provide dimensions of board and k value according to question. We represented the entire code in char 2d arrays format.

We initialized each element with, after, as per user inputs, the value of \* gets updated respectively with X and O.

Input formats-

1. Dimensions of the box
2. Value of k
3. Char X or O( in this code we predefined player 1 as X and player 2 as Y)
4. Player 1 and player 2 have to give position where they want to put until a winner is declared or the game drawn output is displayed( i , j values will be shown in next pages)

Output formats-

1. After giving dimensions board will be displayed as per the given input.
2. Each time board will be shown after a player enters his position
3. In the end, it will show the final panel.

Special Cases-

1. Draw condition- the loop will be continued for m\*n times, and if the itration goes to its max ratio then, the draw will be displayed
2. Invalid move- If a player enters a position that is repeated, then an invalid move will be dispalyed.
3. Invalid move2- If a player enters a position which is not available weak move will be displayed.

Data Structures-

1. 2d character arrays- used character array as the user has to enter X and Y and output will be displayed every time. We intialed 2d array with char ‘\*’ and ‘X’ and ‘O’ will be updated at given places.
2. Function- Code contains 6 main functions they are explained in further slides.
3. For loops- loops are pillars of our code.
4. ASCII values for X and O- Winner declaration is calculated using ascii values of X and O. We find sum and break loops when condition is reached.

These are structures used and each of the following will be explained in next slides.

For an input values mXn, the corresponding I,J values user has to enter are shown in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | J=0 | J=1 | J=2 | J=3 |
| I=0 |  |  |  |  |
| I=1 |  |  |  |  |
| I=2 |  |  |  |  |
| I=3 |  |  |  |  |
| I=4 |  |  |  |  |
| I=5 |  |  |  |  |
| I=6 |  |  |  |  |

Code for 3X3:

I even wrote code for 3X3 grid. Since mXn code explains everything which even includes 3X3 we explained mXn code in this file. We provided code for 3X3 in separate file along with mXn part. We used circular condition for this code. Explaination for mXn is given in next slides.

Main code explanation-

Since the grid is mXn , a for loop is declared for(i=0;i<m\*n;i++). When a player wins break statement breaks for loop and winner will be declared.

We declared all functions in main code with winner conditons

\*\* NOTE- Final code is given in last page.

CORE FUNCTIONS- Our code contains six primary functions to check winning conditions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |

These six core functions will cover the complete grid(mXn).

1. int checkrow(char[][100],int,int,int);
2. int checkcoloumn(char[][100],int,int,int);
3. int checkrupdiagonal(char[][100],int,int,int);
4. int checkrdowndiagonal(char[][100],int,int,int);
5. int checklupdiagonal(char[][100],int,int,int);
6. int checkldowndiagonal(char[][100],int,int,int);

checkrow FUNCTION: Checks every row in the grid(mXn) and if any operator(X,O) appears for k consecutive times that player will be declared as the winner.

EXPLANATION-

We use 3 for loops( 1 for checking each row, 2 for loop is for checking a particular row, 3rd for loop is for checking whether an operator appears for k consecutive times or not i.e winning condition)

This same logic is also used for checkColoumn function.

CODE ( checkrow )

int checkrow(char a[][100],int m,int n,int k)

{

int s,result=0;

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

{

for(int j=0;j<n;j++)

{

s=0;

for(int r=j;r<j+k;r++){

s=s+int(a[i][r]);

if(s==88\*k)

{

result=1;

break;

}

else if(s==79\*k)

{

result=2;

break;

}

}

if(result==1 || result==2)

break;

}

if(result==1 ||result==2)

break;

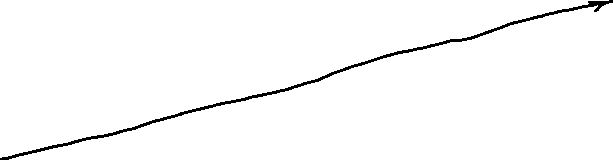
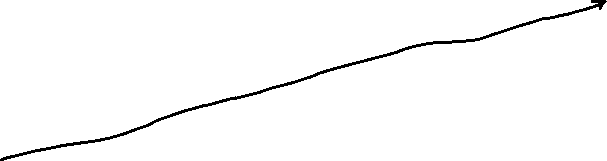
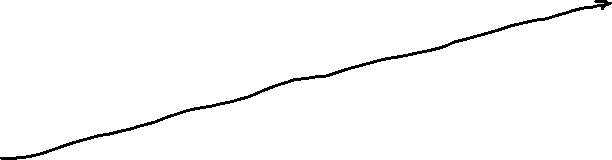
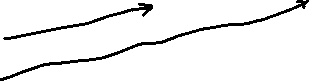
}

return result;

}

This code explains column function too.

Checkrupdiagonal: ( starts form right side(j=0) )



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

EXPLAINATION-

It has 3 for loops. 3rd for loop(inner loop) is for checking winning condition.2nd for loop for stating from j=0 and ending for i=0 and j=n. 1st for loop for checking every diagonal. We calculate s value , if s== 79\*k || s==88\*k we break all the loops and return the value . Winner will be declared in int main() function.

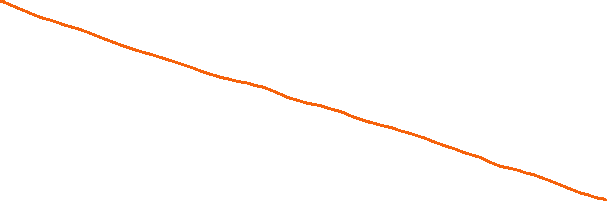
This same logics work for checklup ,checkldown , checkrdown also.

Other Diagonal functions explaination:

checkrdowndiagonal

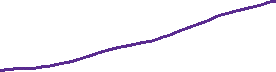
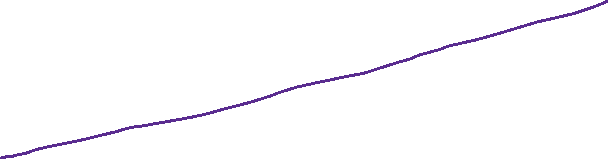
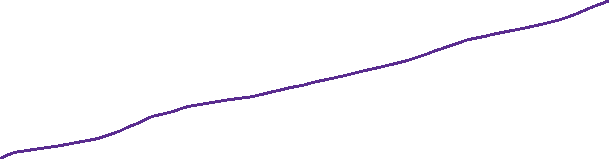
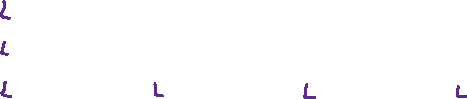
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Checklupdiagonal



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Checkldowndiagonal



|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

This 4 diagrams cover all possible diagonals in the grid which allows us to check each and every winning condition. The logics for all the 4 diagonal funcitons is similar.



Rows and columns functions share similar logic.

CODE for diagonals

int checkrupdiagonal(char a[][100],int m,int n,int k)

{

int i,j,b,p,s,result=0;

for(i=0;i<m;i++)

{

b = i;

for(j=0;j<n;j++)

{

if(b<m)

{

s=0;

for(p=j;p<j+k;p++)

{

s = s+ int(a[b][p]);

b--;

}

if(s==88\*k)

{

result=1;

break;

}

else if(s==79\*k)

{

result=2;

break;

}

b=b+k-1;

}

}

if(result==1 ||result==2)

break;

}

return result;

}

This 2 codes explain all other functions too.

Declaration of the winner- In each and every function an s=0 is initialized and we keep on updating the s value for consecutive k boxes( 3rd for-loop). We break all loops in the inside function if s = 88\*k or 79\*k and we return the s value at the end of the function. In the main code, we declare the winner accordingly.

-Thank you

* Nithin Chepuri(CS21B039)