SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

PROJECT TITLE
Tic-Tac-Toe game

NAME
P. Keerthi Sai Rao

REG NO
RA2111002010015

DEPARTMENT
Mechanical Engineering

SUBMITTED TO

DR. R. RAJKUMAR

DSBS

SCHOOL OF COMPUTING

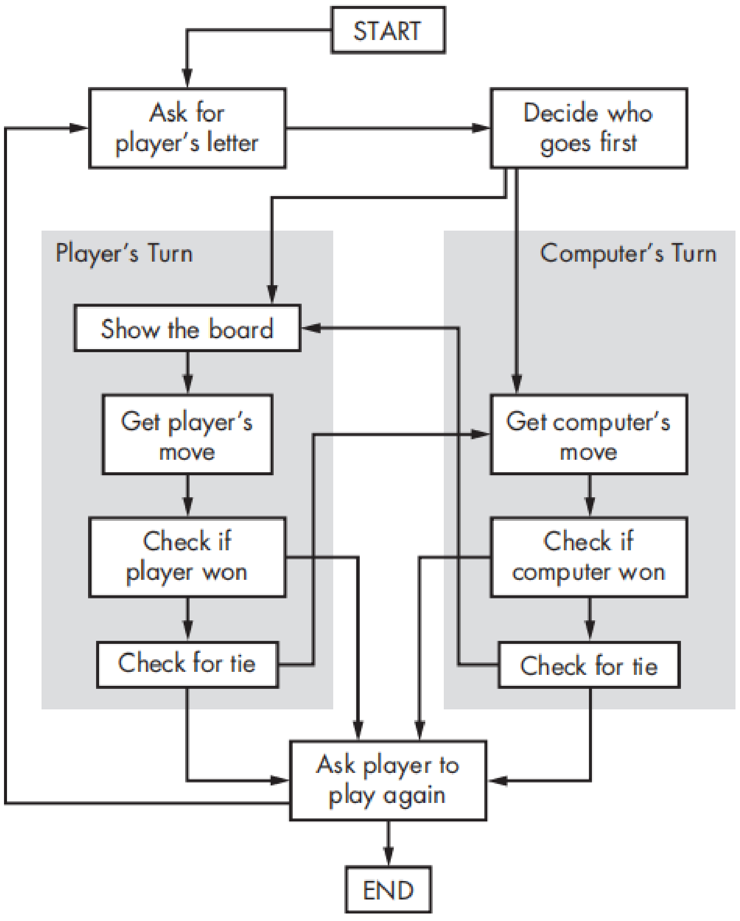
SRMIST

JANUARY 2022

ABSRACT

Tic-Tac-Toe is a fun turn-based game that allows two players to try and match their symbol be it an X or an O in a straight line which can be vertical, horizontal or slanting. The program is not played optimally by both the players because the moves are chosen randomly. The program can easily be modified so that both players play optimally (which will fall under the category of artificial intelligence). The program can also be modified as the user themselves give the input (using scanf() or cin). If both players play optimally then it is only natural that one can’t lose. But they can’t win also since the game will always be tied. It won’t matter who plays first or who plays second, the game will always end up being a draw. While making a Tic Tac Toe game using C language, it is important to make use of arrays. The Xs and Os are kept in different arrays, and they are passed between several functions in the code to keep track of how the game goes. With the code here you can play the game choosing either X or O against the computer. This Tic Tac Toe C game is such that you will have to input a numerical character, from 1 to 9, to select a position for X or O into the space you want. For example: if you are playing with O and you input 2, the O will go to first row – second column. If you want to place O in third row – first column, you have to enter 7. And, it is similar for the other positions. This has been done this way because it is just a console application without graphics designed in C language. The gotoxy function has been used to print text in any part of the screen.

FLOW CHART



PROGRAM

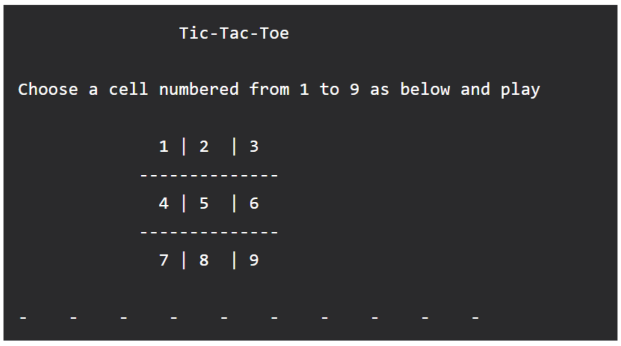
```
// A C++ Program to play tic-tac-toe

#include<bits/stdc++.h>
using namespace std;

#define COMPUTER 1
#define HUMAN 2

#define SIDE 3 // Length of the board

// Computer will move with 'O'
// and human with 'X'
#define COMPUTERMOVE 'O'
#define HUMANMOVE 'X'

// A function to show the current board status
void showBoard(char board[][SIDE])
{
	printf("\n\n");

	printf("\t\t\t %c | %c | %c \n", board[0][0],
							board[0][1], board[0][2]);
	printf("\t\t\t--------------\n");
	printf("\t\t\t %c | %c | %c \n", board[1][0],
							board[1][1], board[1][2]);
	printf("\t\t\t--------------\n");
	printf("\t\t\t %c | %c | %c \n\n", board[2][0],
							board[2][1], board[2][2]);

	return;
}

// A function to show the instructions
void showInstructions()
{
	printf("\t\t\t Tic-Tac-Toe\n\n");
	printf("Choose a cell numbered from 1 to 9 as below"
			" and play\n\n");

	printf("\t\t\t 1 | 2 | 3 \n");
	printf("\t\t\t--------------\n");
	printf("\t\t\t 4 | 5 | 6 \n");
	printf("\t\t\t--------------\n");
	printf("\t\t\t 7 | 8 | 9 \n\n");

	printf("-\t-\t-\t-\t-\t-\t-\t-\t-\n\n");

	return;
}
```

```
// A function to initialise the game
void initialise(char board[][SIDE], int moves[])
{
	// Initiate the random number generator so that
	// the same configuration doesn't arises
	srand(time(NULL));

	// Initially the board is empty
	for (int i=0; i<SIDE; i++)
	{
		for (int j=0; j<SIDE; j++)
			board[i][j] = ' ';
	}

	// Fill the moves with numbers
	for (int i=0; i<SIDE*SIDE; i++)
		moves[i] = i;

	// randomise the moves
	random_shuffle(moves, moves + SIDE*SIDE);

	return;
}

// A function to declare the winner of the game
void declareWinner(int whoseTurn)
{
	if (whoseTurn == COMPUTER)
		printf("COMPUTER has won\n");
	else
		printf("HUMAN has won\n");
	return;
}

// A function that returns true if any of the row
// is crossed with the same player's move
bool rowCrossed(char board[][SIDE])
{
	for (int i=0; i<SIDE; i++)
	{
		if (board[i][0] == board[i][1] &&
			board[i][1] == board[i][2] &&
			board[i][0] != ' ')
			return (true);
	}
	return(false);
}

// A function that returns true if any of the column
// is crossed with the same player's move
bool columnCrossed(char board[][SIDE])
{
	for (int i=0; i<SIDE; i++)
	{
		if (board[0][i] == board[1][i] &&
			board[1][i] == board[2][i] &&
			board[0][i] != ' ')
			return (true);
	}
	return(false);
}

// A function that returns true if any of the diagonal
// is crossed with the same player's move
bool diagonalCrossed(char board[][SIDE])
{
	if (board[0][0] == board[1][1] &&
		board[1][1] == board[2][2] &&
		board[0][0] != ' ')
		return(true);

	if (board[0][2] == board[1][1] &&
		board[1][1] == board[2][0] &&
		board[0][2] != ' ')
		return(true);

	return(false);
}

// A function that returns true if the game is over
// else it returns a false
bool gameOver(char board[][SIDE])
{
	return(rowCrossed(board) || columnCrossed(board)
			|| diagonalCrossed(board) );
}

// A function to play Tic-Tac-Toe
void playTicTacToe(int whoseTurn)
{
	// A 3*3 Tic-Tac-Toe board for playing
	char board[SIDE][SIDE];

	int moves[SIDE*SIDE];

	// Initialise the game
	initialise(board, moves);

	// Show the instructions before playing
	showInstructions();

	int moveIndex = 0, x, y;

	// Keep playing till the game is over or it is a draw
	while (gameOver(board) == false &&
			moveIndex != SIDE*SIDE)
	{
		if (whoseTurn == COMPUTER)
		{
			x = moves[moveIndex] / SIDE;
			y = moves[moveIndex] % SIDE;
			board[x][y] = COMPUTERMOVE;
			printf("COMPUTER has put a %c in cell %d\n",
					COMPUTERMOVE, moves[moveIndex]+1);
			showBoard(board);
			moveIndex ++;
			whoseTurn = HUMAN;
		}

		else if (whoseTurn == HUMAN)
		{
			x = moves[moveIndex] / SIDE;
			y = moves[moveIndex] % SIDE;
			board[x][y] = HUMANMOVE;
			printf ("HUMAN has put a %c in cell %d\n",
					HUMANMOVE, moves[moveIndex]+1);
			showBoard(board);
			moveIndex ++;
			whoseTurn = COMPUTER;
		}
	}

	// If the game has drawn
	if (gameOver(board) == false &&
			moveIndex == SIDE * SIDE)
		printf("It's a draw\n");
	else
	{
		// Toggling the user to declare the actual
		// winner
		if (whoseTurn == COMPUTER)
			whoseTurn = HUMAN;
		else if (whoseTurn == HUMAN)
			whoseTurn = COMPUTER;

		// Declare the winner
		declareWinner(whoseTurn);
	}
	return;
}

// Driver program
int main()
{
	// Let us play the game with COMPUTER starting first
	playTicTacToe(COMPUTER);

	return (0);
}
```

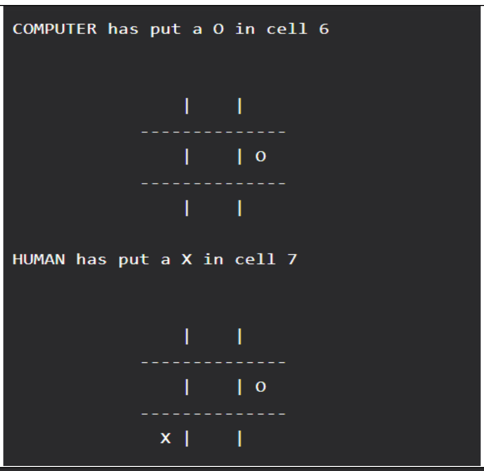
RESULTS

The program is executed and the Tic-Tac-Toe game is playable.

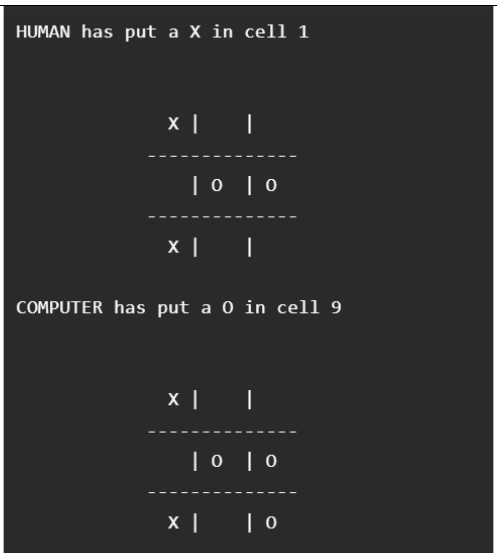
SCREENSHOTS



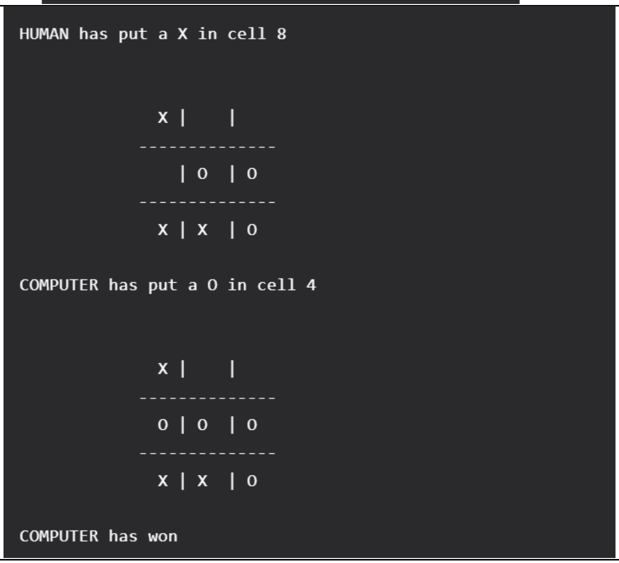
```
                 Tic-Tac-Toe

Choose a cell numbered from 1 to 9 as below and play

              1 | 2  | 3
            -------------
              4 | 5  | 6
            -------------
              7 | 8  | 9

-    -    -    -    -    -    -    -    -    -
```



```
COMPUTER has put a O in cell 6


             |    |
        --------------
             |    | O
        --------------
             |    |

HUMAN has put a X in cell 7


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



```
HUMAN has put a X in cell 1



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

COMPUTER has put a O in cell 9



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



```
HUMAN has put a X in cell 8


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

COMPUTER has put a O in cell 4


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

COMPUTER has won
```

DECLARATION

I would like to thank my parents for everything they have done for me and I would also l like to thank Dr. R. Rajkumar for teaching C and guiding me through the course.

RERERENCES

http://www.codewithc.com
http://www.geeksforgeeks.org