TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A MINI PROJECT ON C PROGRAMMING

"TIC TAC TOE GAME"

Submitted to:

Department of Electronics and

Computer Engineering

Pulchowk Campus

Submitted by:

Neeka Maharjan	077BCT050
Punam Adhikari	077BCT062
Sudip Tiwari	077BCT085
Susheel Thapa	077BCT090

ACKNOWLEDGEMENTS

This is to acknowledge all those without whom this project would not have been reality. Firstly, I would wish to thank our Computer teacher, Ganesh Gautam Sir who gave us opportunity and made us understand how to make this project. Without his guidance, the project would not have been complete.

"Self-belief and hard work will always earn you success." We, the tech-enthusiasts, aspiring to become computer engineers in future, worked really hard and after gaining the useful concepts, we started the project on modular level. Each of our team members shared equal load by writing different parts of the program such as different user-defined functions.

Table of Contents

1. BACKGROUND

For technical courses like computer engineering, it is very important to get a hands-on understanding of our subject and how the theory we learn applies to real-life situations.

So at the end of semester we came to know a lot about c language and we had create a project explaining what we had learnt about c working in the game project "TIC TAC TOE"

This project comes under the curriculum of first semester of Bachelors in Computer engineering.

The project required vast and detailed knowledge and use of conditional statements, loops, structure and file handling functions. We got help from all the lecture classes that happened throughout the semester being held by our computer teacher Mr. Ganesh Gautam. So, to understand and develop the program, we need to learn about the concepts of this language.

2. THEORY

A programming language is the collection of well-structured instructions and keywords that help instruct a computer on what operations to perform.

All programming languages can be divided into two categories:

a. ***Problem oriented languages or High level languages:***

Examples of languages falling in this category are FORTRAN, BASIC, Pascal, COBOL etc. These languages have been designed to give a better programming efficiency i.e. faster program development

b. ***Machine oriented language or Low level languages:***

Examples of languages falling in this category are Assembly language and Machine language.

These languages have been designed to give a better machine efficiency. i.e. faster program execution.

C stands in between these two categories. It is a general purpose high level programming language which was originally developed by Dennis M Ritchie to develop UNIX operating system at bells lab. C is the successor of B language developed at 1970s. C combines the advantages of a high-level language with the functionalism of assembly language.

That's why it is called as Middle level Language, since it was designed to have both; a relatively good programming efficiency (as compared to Machine Oriented Languages) and relatively good machine efficiency (as compared to Problem Oriented Languages).

Compiler, Interpreter and Assembler:

A compiler translates a high level source program into a form that machine can understand. The compiler takes the source code as input and produces the machine language code(object code).

An interpreter, like compiler is also translator which translates high level language into a machine level language. The difference between a compiler and an interpreter is that interpreter translates and executes the program line by line whereas compiler scans the entire program before translating it into machine code.

Structures of C

C is structured programming language. It consists of:

- Pre-processor directives
- Variables and function declaration
- Main function
- Other functions

C uses its character set, reserved or key words, identifier, variables, constants, operators and punctuators.

Control statements:

Logical operation is carried out by several symmetrical or logical statements. There are two types of control statement based on their function.

Selective structure:

Selective structures are used when we have a number of situations where we need to change the order of execution of statements based on certain condition. The selective statements make a decision to take the right path before changing the order of execution. C provides the following statements for selective structure:

If statement:

The if statement first evaluates an expression and then, it executes the statements within its block if the value of the evaluated expression is true. Its syntax is:

```
if(test_expression)

{

Statement-block;

}
```

Nested if statement:

When one if statement is written within the body of another if statement, it is called nested if statement..several if statements can be written as nested if statement. Its syntax is:

```
if(expression)

{

if(expression)

{

Statements

}

}
```

If …else statement:

It is used when there are only two possible options- one happens if a test condition is true and other happens when test condition is false. Its syntax is:

```
if(test_expression)
{
True-block statements;
}
else
{
False-block statements;
}
```

Nested if… else statement:

if….else statements written in the body of another if…else is called nested if…else statement. Its syntax is:

```
if(condition1)
{
    if(condition2)
          {
           Statement 1;
          }
   else
         {
          Statement 2;
         }
else
```

{

Statements;

}

if….else if statement:

An if…else if statement is used when there are more than two possible actions depending upon the outcome of test expression. Its syntax is:

if(condition1)

{

Statement 1;

}

else if(condition 2)

{

Statement 2;

}

else if(condition3)

{

Statement 3;

}

else if(condition_n)

{

Statement n;

}

Switch statements:

When there are number of options available and one of them is to be selected on the basis of some criteria, switch statement is used. It compares a variable or expressions with different constants(int or char type only). Its syntax is:

```
switch(variable or expression)
{
case case_constant1:
    Statements;
    Break;
case  case_constant2:
    Statements;
    Break;

Default:
    Statements;
```

Function:

A function is a self contained program segment that carries out some specific well defined task. Every c program consists of one or functions. Execution of program always begins by carrying out instruction in main. Function makes program significantly easier to understand and maintain. A well written function may be reused in multiple programs. Program that are easier to design, debug and maintain.

Return statement:

A function may or may not send back any value to the calling function. If it does, it is through return statement. The called function can only return only one value per call at most. The syntax of return statement is given below:

```
Return;
```

Arrays

An array is a group of related data items that share a common name. In others words, an array is a data structure that store a number of data items as a single entity (object). The individual data items are called elements and all of them have same data types. Array is used when multiple data

items that have common characteristics are required in a program. In other words, an array is a collection of individual data that is ordered(one can count off the elements 0,1,3..), fixed in size and homogeneous (all elements have to be of the same type e.g., int, float, char,etc).

Pointer:

A pointer is a variable that represents the location (rather than value) of a data item, such as a variable or an array element. Pointers can be used to pass information back and forth between a function and a reference point. Pointer provides a way to return multiple data items from a function via function argument. When a pointer variable is declared, the variable name must be preceded by an aesteric (*).the syntax of a pointer declaration is:

data type *ptar;

Structure:

It is a heterogeneous user defined data type. It is also called constructed data type. It may contain different data types .Structure can also store non homogenous data type into a single collection. Structure may contain pointer, arrays, or even other structures other than the common data types such as int, float, long int etc. A structure provides a means of grouping variables under a single name for easier handling and identification. It can be defined as new named types. It is a convenient way of grouping several pieces of related information together. Complex hierarchies can be created by nesting structures. Structures may be copied to and assigned. They are also useful in passing groups of logically related data into structures. The declaration of structures is given below:

```
struct tag

{
  member 1;
  member 2;

  member n;
};
```

File:

Many applications require that information be written to or read from an auxiliary memory device. Such information is stored on the memory device in the form of a data file. The data files allow us to store information permanently and to access and alter that information whenever necessary.

Opening a file:

Before performing any input / output operation, file must be opened. While opening file, the following must be specified:-

i. The name of file.
ii. The manner in which it should be opened (that for reading ,writing ,both reading and writing ,appending at the end of file, overwriting the file)
iii. when working with a stream oriented data file ,the first step is to establish a buffer area, where information is temporary stored while being transferred between the computers memory and data file .the buffer area is established by writing

FILE *fpt;

where File is a special structure type establishes the buffer area and ptvar is a pointer variable that indicates the beginning of the buffer area the library function fopen is used to open a file .This function is used to open a file .This function is typically written as

fpt=fopen(file name, file type);

where file name and file type are strings that represent the name of the data file and the manner in which the data file will be utilized.

Finally, a file can be closed at the end of the program. This can be accomplished with the library function fclose. The syntax is simply,

fclose(fpt);

Processing a file:

Most data file application requires that a data file be altered as it is being processed. For example, in an application involving the processing of customer records, it may be desirable to add new records to the file. To delete the existing records, to modify the contents or to rearrange the records.

File types:

DOS treats files in two different ways that as binary or text file. Files almost all UNIX systems do not make any distinction between the two. If a file is specified as the binary type, the file I /O functions do not interpret the contents of the file when they read from, or write to the file. But if the file is specified as the text type, the file I / O functions interpret the contents of the file. The basic differences in these two types of files are:

i. ASCII 0*1a is considered as end of file character by the file I /O functions when reading from a text file and they assume that the end of the file has been reached.
ii. In case of DOS, a new line is stored as the sequence 0*0a on the disk in case text files.

3. SOURCE CODE:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <Windows.h>

#define CLEAR_THE_SCREEN system("cls")
#define MAX_LIMIT 100
struct TIME
{
   int seconds;
   int minutes;
};
struct GameData
{
   char player_one[MAX_LIMIT];
   char player_two[MAX_LIMIT];
   int game_status;
   struct TIME time_of_game_play;
   struct TIME start;
   struct TIME stop;
};
char game_array[3][3];
struct GameData data;
SYSTEMTIME str_t;

void gamePage();
void developerInformation();
void welcome();
void howToPlay();
void thankYou();
void loading();
void pressAkeyToContinue();

void gameHomeDisplay();
void gameHomeDisplayOption();
void option();

void doublePlayerMode();
void initializeGameArray();

void takeUserData();
void creatingBoard();
int markingBoard(char[], char);
```

```
void gameStatistics();
void textColorDisplay();
void screenBrightness();

char optionChoice();

int check();

void writeGameData();

void timeOfGamePlay();

int main()
{
   FILE *out_check_data_file;
   char position;
   goto Start;
DoublePlayerMode:
loading();
doublePlayerMode();
printf("\n%10s", "");
if (data.game_status == 1)
   {
      printf("%s WON THE GAME",
data.player_one);
   }
   else if (data.game_status == 2)
   {
      printf("%s WON THE GAME",
data.player_two);
   }
   else if (data.game_status == -1)
   {
      printf("GAME IS DRAW");
   }
pressAkeyToContinue();
writeGameData();
goto Home;
Option:
   loading();
gamePage(" Options ");
option();
position = optionChoice();
switch (position)
   {
   case 'G':
loading();
gamePage("GAME
```

```
STATISTICS");
  gameStatistics();

      pressAkeyToContinue();
goto Option;
break;
case 'T':
loading();
gamePage("TEXT COLOR");
textColorDisplay();
position = optionChoice();
goto TextColor;
break;

   case ('S'):
  loading();
gamePage("SCREEN
BRIGHTNESS");
screenBrightness();
position = optionChoice();
goto ScreenBrightness;
break;
case ('B'):
goto Home;
break;
default:
      loading();

      gamePage("Bad Input");
printf("We are angry with you
because you didnot give us good input
so i won't give you good ouput\n");
printf("Press any key.");
getch();
   goto Option;
break;
   }

TextColor:

   switch (position)
   {
   case '1':
      system("color 1");
      break;
   case '2':
      system("color 2");
      break;
   case '3':
```

```
    system("color 3");
    break;
  case '4':
    system("color 4");
    break;
  case '5':
    system("color 5");
    break;
  case '6':
    system("color 6");
    break;
  case '7':
    system("color 7");
    break;
  case '8':
    system("color 8");
    break;
  case '9':
    system("color 9");
    break;
  case 'A':
    system("color A");
    break;
  case 'B':
    system("color B");
    break;
  case 'C':
    system("color C");
    break;
  case 'E':
    system("color E");
    break;
  case 'F':
    system("color F");
    break;
  default:
    break;
  }
pressAkeyToContinue();
goto Option;
ScreenBrightness:
switch (position)
  {
  case '1':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,10)");
    break;
```

```
case '2':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,20)");
    break;
case '3':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,30)");
    break;
case '4':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,40)");
    break;
  case '5':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,50)");
    break;
case '6':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,60)");
    break;
case '7':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,70)");
    break;
case '8':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,80)");
    break;
  case '9':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,90)");
    break;
case 'A':
    system("powershell (Get-WmiObject -Namespace root/WMI -Class WmiMonitorBrightnessMethods).WmiSetBrightness(1,100)");
```

```
        break;
default:
        break;
    }
    pressAkeyToContinue();

    goto Option;
Start:
out_check_data_file =
fopen("./Data/Statistics.DAT", "r");
if (out_check_data_file == NULL)
    {
        system("attrib +h +s Data");
developerInformation();
pressAkeyToContinue();
     welcome();
pressAkeyToContinue();
    loading();
howToPlay();
pressAkeyToContinue();
fclose(out_check_data_file);
    }
Home:
loading();
gameHomeDisplay();
position = optionChoice();
switch (position)
    {
    case 'D':
goto DoublePlayerMode;
break;
case 'O':
goto Option;
break;
case 'E':
        loading();
gamePage("THANK YOU");
thankYou();
pressAkeyToContinue();
exit(0);
default:
        gamePage("BAD INPUT");
printf("We are angry with you
because you didnot give us good input
so i won't give you good ouput\n");
         pressAkeyToContinue();

        goto Home;
    }
```

```
goto Home;
}
void gamePage(char name[MAX_LIMIT])
{
   "Show the passed string as a heading";
CLEAR_THE_SCREEN;
  printf("\n%13s%-25s\n\n", "", name);
}

void developerInformation()
{
   "This will show the developer information";
gamePage("TIC TAK TOE");
FILE *out_developer;
char ch;
out_developer = fopen("./Data/Developer.txt", "r");
while (1)
   {
      ch = fgetc(out_developer);
      if (ch == EOF)
      {
         break;
      }
      printf("%c", ch);
      Sleep(10);
   }
fclose(out_developer);
}
void welcome()
{
"This will welcome the new user";
gamePage("WELCOME");
FILE *out_welcome;
char ch;
out_welcome = fopen("./Data/Welcome.txt", "r");
while (1)
   {
      ch = fgetc(out_welcome);
      if (ch == EOF)
      {
         break;
      }
      printf("%c", ch);
      Sleep(15);
```

```
    }

    fclose(out_welcome);
}

void howToPlay()
{
    "This will teach user how to play the game";
FILE *out_how_to_play;
char ch;
gamePage("HOW TO PLAY?");
out_how_to_play = fopen("./Data/How_To_Play.txt", "r");
while (1)
    {
        ch = fgetc(out_how_to_play);
        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
        Sleep(5);
    }
    fclose(out_how_to_play);
}

void thankYou()
{
    "This will thanks to the user who had played the game";
    FILE *out_thank_you;
char ch;
int sml = 2;
out_thank_you = fopen("./Data/Thank_You.txt", "r");
while (1)
    {
        ch = fgetc(out_thank_you);
        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
        Sleep(20);
    }
printf("\n\n");
    for (int i = 0; i < 25; i++)
    {
```

```
        printf("%c", sml);
    }
    fclose(out_thank_you);
}

void loading()
{
    "A pattern of loading the data";
int i = 0, j = 0;
CLEAR_THE_SCREEN;
gamePage("");
printf("%15s%-15s\n", "", "Loading");
printf("%15s", "");
while (i < 10)
    {
        Sleep(50);
printf("\xdb");
        i++;
    }
}
void pressAkeyToContinue()
{
    printf("\n\n%10sPress a key to continue", "");
    getch();
}
void gameHomeDisplay()
{
    "This will show the home page of screen with option";

    gamePage(" TIC TAK TOE");
printf("\n%10s%-30s", "", "Double Player Mode");
printf("\n%10s%-30s", "", "Option");
printf("\n%10s%-30s", "", "Exit");
}
void option()
{
    "Show option for option in Home page";
printf("\n%10s%-30s", "", "Game Statistics");
printf("\n%10s%-30s", "", "Text Color");
 printf("\n%10s%-30s", "", "Screen Brightness");
printf("\n%10s%-30s", "", "Back");
}
char optionChoice()
```

```
{
   "This will let user to position a option and return the choiced option";
char position;
printf("\n\n%8s%-20s", " ", "Choose any one of the above?\n");
printf("%5s-->", " ");
  fflush(stdin); /*This line is important*/
position = getchar();
position = toupper(position);
        return position;
}

void doublePlayerMode()
{
   "Double Player Game Mode";
   char position[3], player_symbol;
   int player = 1, mark_status = 1;
initializeGameArray();
takeUserData();
do
   {
   TOP:
      gamePage("DOUBLE PLAYER MODE");
creatingBoard();
player = (player % 2) ? 1 : 2;

 (player == 1) ? printf("%s\nPosition: ", data.player_one) : printf("%s\nPosition: ", data.player_two);
      scanf("%s", &position);

      player_symbol = (player == 1) ? 'X' : 'O';

      mark_status = markingBoard(position, player_symbol);

      if (mark_status == 0)
      {
         printf("\n%10sIt has be already choosen\n", "");
        pressAkeyToContinue();
         goto TOP;
      }
      data.game_status = check();
```

```
    player++;
    } while (data.game_status == 0);

gamePage("DOUBLE PLAYER MODE");
creatingBoard();
    GetSystemTime(&str_t);

  data.stop.minutes = str_t.wMinute;
  data.stop.seconds = str_t.wSecond;
}

void initializeGameArray()
{
  for (int i = 0; i < 3; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      game_array[i][j] = ' ';
    }
  }
}
void takeUserData()
{
  gamePage("TAKING USER INPUT");
   printf("%10sPlayer One: ", "");
  scanf(" %[^\n]", data.player_one);
  printf("\n");
 printf("%10sPlayer Two: ", "");
  scanf(" %[^\n]", data.player_two);

  data.game_status = 0;
 printf("\n%10sPress any key to start the game", "");
getch();
GetSystemTime(&str_t);
  data.start.minutes = str_t.wMinute;
  data.start.seconds = str_t.wSecond;
}
void creatingBoard()
{
  "Display TIC TAK TOE Board";
  printf("%8s%s(X) vrs %s(O) \n\n", "", data.player_one, data.player_two);

  printf("%8s     |     |     ", "");
  printf("%10s     |     |     \n", "");
```

```
   printf("%8s  %c  |  %c  |  %c ",
"", game_array[0][0],
game_array[0][1],
game_array[0][2]);

   printf("%10s  %s  |  %s  |  %s
\n", "", "NW", "NN", "NE");


   printf("%8s      |      |      ", "");

   printf("%10s      |      |      \n", "");


   printf("%8s----------------------",
"");

   printf("%10s----------------------
\n", "");


   printf("%8s      |      |      ", "");

   printf("%10s      |      |      \n", "");


   printf("%8s  %c  |  %c  |  %c ",
"", game_array[1][0],
game_array[1][1],
game_array[1][2]);

   printf("%10s  %s  |  %s  |  %s
\n", "", "CW", "CC", "CE");

   printf("%8s      |      |      ", "");

   printf("%10s      |      |      \n", "");


   printf("%8s----------------------",
"");

   printf("%10s----------------------
\n", "");


   printf("%8s      |      |      ", "");

   printf("%10s      |      |      \n", "");


   printf("%8s  %c  |  %c  |  %c ",
"", game_array[2][0],
game_array[2][1],
game_array[2][2]);

   printf("%10s  %s  |  %s  |  %s
\n", "", "SW", "SS", "SE");


   printf("%8s      |      |      ", "");

   printf("%10s      |      |      \n\n",
"");

}


int markingBoard(char position[],
char player)

{

   "Update Character Array";


   if (strcmpi(position, "NW") == 0)

   {
```

```
    if (game_array[0][0] != ' ')
    {
        return 0;
    }
    game_array[0][0] = player;
}
else if (strcmpi(position, "NN") ==
0)
{
    if (game_array[0][1] != ' ')
    {
        return 0;
    }
    game_array[0][1] = player;
}
else if (strcmpi(position, "NE") ==
0)
{
    if (game_array[0][2] != ' ')
    {
        return 0;
    }
    game_array[0][2] = player;
}
else if (strcmpi(position, "CW") ==
0)
{
    if (game_array[1][0] != ' ')
    {
        return 0;
    }
    game_array[1][0] = player;
}
else if (strcmpi(position, "CC") ==
0)
{
    if (game_array[1][1] != ' ')
    {
        return 0;
    }
    game_array[1][1] = player;
}
else if (strcmpi(position, "CE") ==
0)
{
    if (game_array[1][2] != ' ')
    {
        return 0;
    }
```

```
        game_array[1][2] = player;
    }
    else if (strcmpi(position, "SW") ==
0)
    {
        if (game_array[2][0] != ' ')
        {
            return 0;
        }
        game_array[2][0] = player;
    }
    else if (strcmpi(position, "SS") ==
0)
    {
        if (game_array[2][1] != ' ')
        {
            return 0;
        }
        game_array[2][1] = player;
    }
    else if (strcmpi(position, "SE") ==
0)
    {
        if (game_array[2][2] != ' ')
        {
            return 0;
        }
        game_array[2][2] = player;
    }

    return 1;
}
int check()
{
    "Check whether anyone has won or
not";
    if (game_array[1][1] == 'X')
    {
        if ((game_array[0][0] == 'X'
&& game_array[2][2] == 'X') ||
(game_array[0][2] == 'X' &&
game_array[2][0] == 'X') ||
(game_array[0][1] == 'X' &&
game_array[2][1] == 'X') ||
(game_array[1][0] == 'X' &&
game_array[1][2] == 'X'))
        {
            return 1;
        }
    }
    if (game_array[0][0] == 'X')
    {
```

```
    if ((game_array[1][0] == 'X' && game_array[2][0] == 'X') || (game_array[0][1] == 'X' && game_array[0][2] == 'X'))
    {
        return 1;
    }
}

if (game_array[2][2] == 'X')
{
    if ((game_array[0][2] == 'X' && game_array[1][2] == 'X') || (game_array[2][0] == 'X' && game_array[2][1] == 'X'))
    {
        return 1;
    }
}

if (game_array[1][1] == 'O')
{
    if ((game_array[0][0] == 'O' && game_array[2][2] == 'O') || (game_array[0][2] == 'O' && game_array[2][0] == 'O') || (game_array[0][1] == 'O' && game_array[2][1] == 'O') || (game_array[1][0] == 'O' && game_array[1][2] == 'O'))
    {
        return 2;
    }
}

if (game_array[0][0] == 'O')
{
    if ((game_array[1][0] == 'O' && game_array[2][0] == 'O') || (game_array[0][1] == 'O' && game_array[0][2] == 'O'))
    {
        return 2;
    }
}

if (game_array[2][2] == 'O')
{
    if ((game_array[0][2] == 'O' && game_array[1][2] == 'O') || (game_array[2][0] == 'O' && game_array[2][1] == 'O'))
    {
        return 2;
    }
```

```
    }


    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (game_array[i][j] == ' ')
            {
                return 0;
            }
        }
    }


    return -1;
}


void timeOfGamePlay()
{


    data.time_of_game_play.minutes =
data.stop.minutes -
data.start.minutes;

    data.time_of_game_play.seconds =
data.stop.seconds -
data.start.seconds;

    if
(data.time_of_game_play.seconds <
0)
    {
        data.time_of_game_play.seconds
= data.time_of_game_play.seconds +
60;
        data.time_of_game_play.minutes
= data.time_of_game_play.minutes -
1;
    }
}


void writeGameData()
{
    FILE *in_game_data;


    in_game_data =
fopen("./Data/Statistics.DAT", "a");


    if (in_game_data == NULL)
    {
        return;
    }


    timeOfGamePlay();
```

```
    fprintf(in_game_data, "%-15s%-20s%-20s %5dsec %s%s\n", __DATE__, data.player_one, data.player_two, data.time_of_game_play.minutes * 60 + data.time_of_game_play.seconds, (data.game_status == -1) ? ("Draw") : ((data.game_status == 1) ? (data.player_one) : (data.player_two)), (data.game_status != -1) ? (" won") : (" "));


    fclose(in_game_data);
}


void gameStatistics()
{
    FILE *out_game_data;
    char ch;


    out_game_data = fopen("./Data/Statistics.DAT", "r");


    if (out_game_data == NULL)
    {
        printf("%10sSome error occured while retrieving the data.\n", "");
        printf("%10sPlease try again later\n", "");
        printf("%10sIf it doesn't work try restarting the game.\n", "");
        return;
    }


    printf("\n\n%-15s%-20s%-20s%-20s%-30s\n\n", "DATE", "PLAYER ONE", "PLAYER TWO", "TIME OF GAME PLAY", "RESULT");
    while (1)
    {
        ch = fgetc(out_game_data);


        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
    }


    fclose(out_game_data);
}


void textColorDisplay()
{
    "Option for Text Color";
```

```
   printf("%-15s%10s%-15s\n", "1 =
Blue  ", "", "9 = Light Blue  ");

   printf("%-15s%10s%-15s\n", "2 =
Green ", "", "A = Light Green ");

   printf("%-15s%10s%-15s\n", "3 =
Aqua  ", "", "B = Light Aqua  ");

   printf("%-15s%10s%-15s\n", "4 =
Red   ", "", "C = Light Red   ");

   printf("%-15s%10s%-15s\n", "5 =
Purple", "", "D = Light Purple");

   printf("%-15s%10s%-15s\n", "6 =
Yellow", "", "E = Light Yellow");

   printf("%-15s%10s%-15s\n", "7 =
White ", "", "F = Bright White");

   printf("%-15s%10s%-15s\n", "8 =
Gray  ", "", "");

}


void screenBrightness()

{

   "Option for screen Display";


   printf("%10s%-15s\n", "", " 1 =
10% Brightness ");

   printf("%10s%-15s\n", "", " 2 =
20% Brightness ");
```

```
 printf("%10s%-15s\n", "", " 3 =  30% Brightness ");
   printf("%10s%-15s\n", "", " 4 =  40% Brightness ");
   printf("%10s%-15s\n", "", " 5 =  50% Brightness ");
   printf("%10s%-15s\n", "", " 6 =  60% Brightness ");
   printf("%10s%-15s\n", "", " 7 =  70% Brightness ");
   printf("%10s%-15s\n", "", " 8 =  80% Brightness ");
   printf("%10s%-15s\n", "", " 9 =  90% Brightness ");
   printf("%10s%-15s\n", "", " A = 100% Brightness ");
}
```

4. DISCUSSION

A. Introduction to game

The tic-tac-toe game is played on a 3x3 grid the game is played by two players, who take turns. The first player marks moves with a circle, the second with a cross. The player who has formed a horizontal, vertical, or diagonal sequence of three marks wins. The program should draws the game board, asks the user for the coordinates of the next mark, change the players after every successful move, and announce the winner.

B. User Defined Functions

Function Name: gamePage

```
void gamePage(char name[MAX_LIMIT])
{
    "Show the passed string as a heading";

    CLEAR_THE_SCREEN;

    printf("\n%13s%-25s\n\n", "", name);
}
```

This function takes a string and display it on the string in a formatted way.

Function Name: developerInformation

```
void developerInformation()
{
    "This will show the developer information";

    gamePage("TIC TAK TOE");

    FILE *out_developer;

    char ch;

    out_developer = fopen("./Data/Developer.txt", "r");

    while (1)
    {
        ch = fgetc(out_developer);
        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
        Sleep(10);
    }

    fclose(out_developer);
}
```

This function open a file “Developer.txt” in read mode present in Data folder and display all the content of the folder.

To print the content of files we have use character I/O.

Also we have used Sleep(10) to freeze our program for 10 millisecond which have huge effect when program is executed.

Function Name: Welcome

```
void welcome()
{
    "This will welcome the new user";

    gamePage("WELCOME");

    FILE *out_welcome;

    char ch;

    out_welcome = fopen("./Data/Welcome.txt", "r");

    while (1)
    {
        ch = fgetc(out_welcome);

        if (ch == EOF)
        {
            break;
        }

        printf("%c", ch);

        Sleep(15);
    }

    fclose(out_welcome);
}
```

This function open a file “Welcome.txt” in read mode present in Data folder and display all the content of the folder.

To print the content of files we have use character I/O.

Also we have used Sleep(15) to freeze our program for 15 millisecond which have huge effect when program is executed.

Function Name: howtoPlay

```
void howToPlay()
{
    "This will teach user how to play the game";

    FILE *out_how_to_play;

    char ch;

    gamePage("HOW TO PLAY?");

    out_how_to_play = fopen("./Data/How_To_Play.txt", "r");

    while (1)
    {
        ch = fgetc(out_how_to_play);
        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
        Sleep(5);
    }

    fclose(out_how_to_play);
}
```

This function open a file “How_To_Play.txt” in read mode present in Data folder and display all the content of the folder.

To print the content of files we have use character I/O.

Also we have used Sleep(5) to freeze our program for 5 millisecond which have huge effect when program is executed.

Function Name: Loading

```
void loading()
{
    "A pattern of loading the data";

    int i = 0, j = 0;

    CLEAR_THE_SCREEN;

    gamePage("");

    printf("%15s%-15s\n", "", "Loading");

    printf("%15s", "");

    while (i < 10)
    {
        Sleep(50);

        printf("\xdb");

        i++;
    }
}
```

This function is responsible to display a pattern while we navigate from one page to another to tell user that we are redirecting to that page.

This function is called when we are redirecting from one page to another.

Function Name: pressAKeyToContinue

```
void pressAkeyToContinue()
{
    "Ask user a key to press to move ahead";

    printf("\n\n%10sPress a key to continue", "");

    getch();
}
```

This function display a message to user, “Press a key to continue.”

And it accept the press key.

This function is called when some process is finished and to tell user that the process that was running had come to end so you can proceed ahead.

Function Name: badInput

```
void badInput()
{
    "Display the bad input information to the user";

    gamePage("BAD INPUT");

    printf("%10sPlease try to given valid input.\n", "");

    printf("\n%10sTry again with valid input.\n", " ");
}
```

This function is used to display a message when a user doesn’t give a valid input.

This function is called when user is likely to give wrong input.

Function Name: gameHomeDisplay

```
void gameHomeDisplay()
{
    "This will show the home page of screen with option";

    gamePage(" TIC TAK TOE");

    printf("\n%10s%-30s", "", "Double Player Mode");
    printf("\n%10s%-30s", "", "Option");
    printf("\n%10s%-30s", "", "Exit");
}
```

This function is used to display the option that a user has while playing the game.

Function Name: option

```
void option()
{
    "Show option for option in Home page";

    printf("\n%10s%-30s", "", "Game Statistics");
    printf("\n%10s%-30s", "", "Text Color");
    printf("\n%10s%-30s", "", "Screen Brightness");
    printf("\n%10s%-30s", "", "Back");
}
```

This function works when the user enters 2 inside void gameHomeDisplay(). As mentioned above, its major function is to make the game more user friendly and customized. Here,user can select:

1. Game statistics
2. Text Colour
3. Screen Brightness
4. How to play?
5. Developer
6. Back

The option "Game statistics" lets the user view all the previous game stats in their computer.

Likewise, "Text Colour" helps them to adjust various types of text colours. Similarly, "Screen Brightness " is for adjustment of brightness of the game display and "How to Play?" option assists user to learn how to play the game. "Developer" Is to provide developer information to the user and finally "Back" option takes them to our initial void gameHomeDisplay() option.

Function Name: DoublePlayerMode

```
void doublePlayerMode()
{
    "Double Player Game Mode";

    char position[3], player_symbol;
    int player = 1, mark_status = 1;

    initializeGameArray();

    takeUserData();

    do
    {
    TOP:
        gamePage("DOUBLE PLAYER MODE");

        creatingBoard();

        player = (player % 2) ? 1 : 2;

        (player == 1) ? printf("%s\nPosition: ", data.player_one) : printf("%s\nPosition: ",
        data.player_two);
        scanf("%s", &position);

        player_symbol = (player == 1) ? 'X' : 'O';

        mark_status = markingBoard(position, player_symbol);

        if (mark_status == 0)
        {
            printf("\n%10sIt may have been already choosen or the option doesnot exist.\n",
            "");

            pressAkeyToContinue();

            goto TOP;
        }
        data.game_status = check();

        player++;

    } while (data.game_status == 0);

    gamePage("DOUBLE PLAYER MODE");

    creatingBoard();

    GetSystemTime(&str_t);

    data.stop.minutes = str_t.wMinute;
    data.stop.seconds = str_t.wSecond;
}
```

This uses some local variable to run properly.

Then it called **intializeGameArray()** which will make all the element of game array to " "(space).

Then it calls **takeUserData()** which is responsible to take data of user who player the game.

Then a loop is executed.

Loop:

- This loop is executed till gamestatus is equal to 0 which indicates that game has not finished
- Then it call gamePage(discussed above)
- Then it call **creatingBoard()** which will create game board
- Then player will is alternate i.e it switch user 0 for on user and 1 for next user
- On the basis of above, It ask the respective user to input and update player_symbol.
- Then it pass the data to the function **markingBoard()** which will update the game board.
- Return value of markingBoard is analyses and proceed according to it.
  Value 0 means so error has occurred.

  So when value equal to zero it display the message that error has occurred and direct our flows of program to Top of loop so that player, board won't be update and player can give correct data this time

- If error hasn't occurred then it call check() to check if anyone has won the game or not
- And return value is will determine whether the loop will executed or not

Then it call creatingBoard function to display the board once again. Then below code will record the time when game has ended.

Function Name: InitializeGameArray

```
void initializeGameArray()
{
    "Make all the element of the game array to space";

    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            game_array[i][j] = ' ';
        }
    }
}
```

It just iterate over every element of game array and place “ ” in every element.

Function Name: takeUserData

```
void takeUserData()
{
    "This takes user data like player one name, player two name";

    gamePage("TAKING USER INPUT");

    printf("%10sPlayer One: ", "");
    scanf(" %[^\n]", data.player_one);

    printf("\n");

    printf("%10sPlayer Two: ", "");
    scanf(" %[^\n]", data.player_two);

    data.game_status = 0;

    printf("\n%10sPress any key to start the game", "");

    getch();

    GetSystemTime(&str_t);

    data.start.minutes = str_t.wMinute;
    data.start.seconds = str_t.wSecond;
}
```

This takes the name of player one, player two and Time when the game has started.

This set game status to zero indicating game is going on between to player.

This is called inside DoublePlayerFunction as mention above.

Function Name: createBoard()

```
void creatingBoard()
{
    "Display TIC TAK TOE Board";

    printf("%8s%s(X) vrs %s(O) \n\n", "", data.player_one, data.player_two);

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n", "");

    printf("%8s    %c   |   %c   |   %c  ", "", game_array[0][0], game_array[0][1], game_array[0][2]);
    printf("%10s   %s   |  %s   |  %s  \n", "", "NW", "NN", "NE");

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n", "");

    printf("%8s————————————————————", "");
    printf("%10s————————————————————\n", "");

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n", "");

    printf("%8s    %c   |   %c   |   %c  ", "", game_array[1][0], game_array[1][1], game_array[1][2]);
    printf("%10s   %s   |  %s   |  %s  \n", "", "CW", "CC", "CE");

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n", "");

    printf("%8s————————————————————", "");
    printf("%10s————————————————————\n", "");

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n", "");

    printf("%8s    %c   |   %c   |   %c  ", "", game_array[2][0], game_array[2][1], game_array[2][2]);
    printf("%10s   %s   |  %s   |  %s  \n", "", "SW", "SS", "SE");

    printf("%8s        |       |      ", "");
    printf("%10s        |       |      \n\n", "");
}
```

This function create a board with all the marks that user has chosen.

Function Name: markingBoard

```
int markingBoard(char position[], char player)
{
    "Update Character Array";

    if (strcmpi(position, "NW") == 0 || strcmpi(position, "NN") == 0 || strcmpi(position, "NE") == 0 || strcmpi(position, "CW") == 0 || strcmpi(position, "CC") == 0 ||
    strcmpi(position, "CE") == 0 || strcmpi(position, "SW") == 0 || strcmpi(position, "SS") == 0 || strcmpi(position, "SE") == 0)
    {
        if (strcmpi(position, "NW") == 0)
        {
            if (game_array[0][0] != ' ')
            {
                return 0;
            }
            game_array[0][0] = player;
        }
        else if (strcmpi(position, "NN") == 0)
        {
            if (game_array[0][1] != ' ')
            {
                return 0;
            }
            game_array[0][1] = player;
        }
        else if (strcmpi(position, "NE") == 0)
        {
            if (game_array[0][2] != ' ')
            {
                return 0;
            }
            game_array[0][2] = player;
        }
        else if (strcmpi(position, "CW") == 0)
        {
            if (game_array[1][0] != ' ')
            {
                return 0;
            }
            game_array[1][0] = player;
        }
        else if (strcmpi(position, "CC") == 0)
        {
            if (game_array[1][1] != ' ')
            {
                return 0;
            }
            game_array[1][1] = player;
        }
        else if (strcmpi(position, "CE") == 0)
        {
            if (game_array[1][2] != ' ')
            {
                return 0;
            }
            game_array[1][2] = player;
        }
        else if (strcmpi(position, "SW") == 0)
        {
            if (game_array[2][0] != ' ')
            {
                return 0;
            }
            game_array[2][0] = player;
        }
        else if (strcmpi(position, "SS") == 0)
        {
            if (game_array[2][1] != ' ')
            {
                return 0;
            }
            game_array[2][1] = player;
        }
        else if (strcmpi(position, "SE") == 0)
        {
            if (game_array[2][2] != ' ')
            {
                return 0;
            }
            game_array[2][2] = player;
        }

        return 1;
    }

    else
    {
        return 0;
    }
}
```

This function takes two argument position(where player want to mark the game) and player(who has choose that position)

It analysis the position and update game board.

If the position doesn't exist or it has been already occupied then it return 0 indicating error. If not it return 1 by updating board.

Function Name: gameStatistics

```
void gameStatistics()
{
    "Disply all the game record that has been stored in our data base";

    FILE *out_game_data;
    char ch;

    out_game_data = fopen("./Data/Statistics.DAT", "r");

    if (out_game_data == NULL)
    {
        printf("%10sSome error occured while retrieving the data.\n", "");
        printf("%10sPlease try again later\n", "");
        printf("%10sIf it doesn't work try restarting the game.\n", "");
        return;
    }

    printf("\n\n%-15s%-20s%-20s%-20s%-30s\n\n", "DATE", "PLAYER ONE", "PLAYER
    TWO", "TIME OF GAME PLAY", "RESULT");
    while (1)
    {
        ch = fgetc(out_game_data);

        if (ch == EOF)
        {
            break;
        }
        printf("%c", ch);
    }

    fclose(out_game_data);
}
```

This function displays the content of file "Statistics.DAT" present inside Data folder in a tabular form

We have use character I/O to read and write to the file

Function Name: option

```
void option()
{
    "Show option for option in Home page";

    printf("\n%10s%-30s", "", "Game Statistics");
    printf("\n%10s%-30s", "", "Text Color");
    printf("\n%10s%-30s", "", "Screen Brightness");
    printf("\n%10s%-30s", "", "Back");
}
```

This function display all the option present in the Option of ***gameHomeDisplay Function***

Function Name: TextColorDisplay

```
void textColorDisplay()
{
    "Option for Text Color";

    printf("%-15s%10s%-15s\n", "1 = Blue  ", "", "9 = Light Blue  ");
    printf("%-15s%10s%-15s\n", "2 = Green ", "", "A = Light Green ");
    printf("%-15s%10s%-15s\n", "3 = Aqua  ", "", "B = Light Aqua  ");
    printf("%-15s%10s%-15s\n", "4 = Red   ", "", "C = Light Red   ");
    printf("%-15s%10s%-15s\n", "5 = Purple", "", "D = Light Purple");
    printf("%-15s%10s%-15s\n", "6 = Yellow", "", "E = Light Yellow");
    printf("%-15s%10s%-15s\n", "7 = White ", "", "F = Bright White");
    printf("%-15s%10s%-15s\n", "8 = Gray  ", "", "");
}
```

This function displays all the option available under Text color option in Option.

Function Name: screenBrightness

```
void screenBrightness()
{
    "Option for screen Display";

    printf("%10s%-15s\n", "", " 1 =  10% Brightness ");
    printf("%10s%-15s\n", "", " 2 =  20% Brightness ");
    printf("%10s%-15s\n", "", " 3 =  30% Brightness ");
    printf("%10s%-15s\n", "", " 4 =  40% Brightness ");
    printf("%10s%-15s\n", "", " 5 =  50% Brightness ");
    printf("%10s%-15s\n", "", " 6 =  60% Brightness ");
    printf("%10s%-15s\n", "", " 7 =  70% Brightness ");
    printf("%10s%-15s\n", "", " 8 =  80% Brightness ");
    printf("%10s%-15s\n", "", " 9 =  90% Brightness ");
    printf("%10s%-15s\n", "", " A = 100% Brightness ");
}
```

This function displays all the option available under Screen Brightness option in Option.

Function name: optionChoice

```
char optionChoice()
{
    "This will let user to position a option and return the choiced option";

    char position;

    printf("\n\n%8s%-20s", " ", "Choose any one of the above?\n");
    printf("%5s⟶", " ");

    fflush(stdin); /*This line is important*/

    position = getchar();

    position = toupper(position);

    return position;
}
```

This function tell use to choice any option when option is display and it return the character the user have choose This function is called whenever we give user to choice option out of things we have displays fflush(stdin) → Clear Buffer in the RAM

Function name: check

```
int check()
{
    "Check whether anyone has won or not";

    if (game_array[1][1] == 'X')
    {
        if ((game_array[0][0] == 'X' && game_array[2][2] == 'X') || (game_array[0][2] == 'X' && game_array[2][0] == 'X')
        || (game_array[0][1] == 'X' && game_array[2][1] == 'X') || (game_array[1][0] == 'X' && game_array[1][2] == 'X'))
        {
            return 1;
        }
    }
    if (game_array[0][0] == 'X')
    {
        if ((game_array[1][0] == 'X' && game_array[2][0] == 'X') || (game_array[0][1] == 'X' && game_array[0][2] == 'X'))
        {
            return 1;
        }
    }

    if (game_array[2][2] == 'X')
    {
        if ((game_array[0][2] == 'X' && game_array[1][2] == 'X') || (game_array[2][0] == 'X' && game_array[2][1] == 'X'))
        {
            return 1;
        }
    }
```

```
    if (game_array[1][1] == 'O')
    {
        if ((game_array[0][0] == 'O' && game_array[2][2] == 'O') || (game_array[0][2] == 'O' && game_array[2][0] == 'O') ||
        (game_array[0][1] == 'O' && game_array[2][1] == 'O') || (game_array[1][0] == 'O' && game_array[1][2] == 'O'))
        {
            return 2;
        }
    }

    if (game_array[0][0] == 'O')
    {
        if ((game_array[1][0] == 'O' && game_array[2][0] == 'O') || (game_array[0][1] == 'O' && game_array[0][2] == 'O'))
        {
            return 2;
        }
    }

    if (game_array[2][2] == 'O')
    {
        if ((game_array[0][2] == 'O' && game_array[1][2] == 'O') || (game_array[2][0] == 'O' && game_array[2][1] == 'O'))
        {
            return 2;
        }
    }
```

```
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (game_array[i][j] == ' ')
            {
                return 0;
            }
        }
    }

    return -1;
}
```

The game of Tic Tac Toe is a multiplayer, multistep game meaning the game program needs to check the condition of winning, consequently losing, similarly, continuing and drawing of the game every time a player makes his/her move. Think of it like this: you are playing the game with your friend. While doing so, your mind subconsciously checks whether or not your move has caused the game to finish either by your victory or opponent's. Same is the case here, except the fact that a function namely Check 'check()' examines if the move by any one of the two players leads the game to end or not. Comprehensively, the check function checks the characters 'X', 'O' and ' ' in each row, column or cross possible and returns value necessarily. If Player 1 'X' enters his choice then this function instantly scans if that move has caused the game to be won by Player 1. If yes, then it returns value 1 to the board function mentioned above. Likewise, if Player 2 'O' enters his choice then it immediately scrutinizes if the move has resulted a win to Player 2 or not. If yes, it returns value 2 to the board.

A game is continued unless this function returns 1 or 2 or -1. When no one has won the game yet but any one of the nine slots is empty ' ', this function returns the value 0 to board function which in return asks another player to enter his choice. Else, when no player has succeeded and none of the position is free, the function returns -1 which displays that the game has ended in draw.

Function name: writeGameData

```
void writeGameData()
{
    "Write the game data to the file";

    FILE *in_game_data;

    in_game_data = fopen("./Data/Statistics.DAT", "a");

    if (in_game_data == NULL)
    {
        return;
    }

    timeOfGamePlay();

    fprintf(in_game_data, "%-15s%-20s%-20s      %5dsec      %s%s\n", __DATE__, data.player_one, data.
    player_two, data.time_of_game_play.minutes * 60 + data.time_of_game_play.seconds, (data.game_status ==
    -1) ? ("Draw") : ((data.game_status == 1) ? (data.player_one) : (data.player_two)), (data.game_status
    != -1) ? (" won") : (" "));

    fclose(in_game_data);
}
```

This function open a file called “Statistics.DAT” present in Data folder I append mode.

It call timeofGamePlay which evaluate the total time of game play.

Then, it just write all the data to the file like Data when the game was played, Player’s name,Who had won the game and Time of their game play.

Here, we have used concept of formatted I/O.

Function name: timeOfGamePlay

```
void timeOfGamePlay()
{
    "Calculate time of game play between two player";

    data.time_of_game_play.minutes = data.stop.minutes - data.start.minutes;
    data.time_of_game_play.seconds = data.stop.seconds - data.start.seconds;
        struct GameData data
    if (data.time_of_game_play.seconds < 0)
    {
        data.time_of_game_play.seconds = data.time_of_game_play.seconds + 60;
        data.time_of_game_play.minutes = data.time_of_game_play.minutes - 1;
    }
}
```

Since we have store the time of start of game play and stop of game play.

This function basically subtract the time when game ended and start and store the result in

timeofgameplay is defined in struct variable data.

c. Editor, Compiler, and Machine we choose to work

C Editor

Here, we have used Visual Studio Code as my editor. You can download the editor from Download Visual Studio Code - Mac, Linux, Windows . Select your operating system and download it.

Compiler

Here, we have used **gcc** as my compiler provided by MinGWw64. You can download it via Download MinGW-w64 - for 32 and 64 bit Windows from SourceForge.net. Your download will start automatically. Run the downloaded .exe file. After, you have install MinGW-w64, you need to configure it.

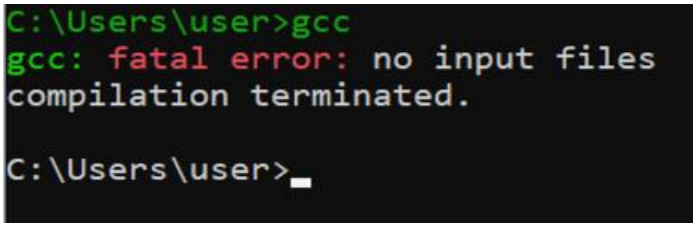
1. In the Windows search bar, type 'settings' to open your Windows Settings.
2. Search for Edit environment variables for your account.
3. Choose the Path variable and then select Edit.
4. Select New and add the Mingw-w64 destination folder path to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If you used the settings above to install Mingww64, then add this to the path: **C:\Program Files\mingw-w64\x86_64-8.1.0posix-seh-rt_v6-rev0\mingw64\bin.**

5. Select OK to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available.

Check your installation

Open command prompt or powershell and type:

```
C:\Users\user>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```



If you get similar result, you are good to go.

MACHINE WE CHOOSE TO WORK

Here,we used our windows operating system for completing our c proect in tic tac toe.

5. COMPILATION STAGES

There are basically four steps in compilation of C Program:

- Pre-processing
- Compilation
- Assembly
- Linking

Sample Program:

```
#include

<stdio.h>void

main()

{

        printf(“Hello World”);

}
```

Pre-processing

- First stage of compilation
- Line starting with a # character are resolved
- Joined continued lines(line ending with a \) and stripping command

To see pre-processing stage, pass -E to option in

gcc:gcc -E Hello.c

This will display the pre-processed files in the terminal.

If you look at the pre-processed files you will see that all the # commnad are resolve along with comment are removed and macro are resolved

Compilation

Pre Processed code is translated to assembly instruction to the target processor achiteture
It form intermediate human readable language

To see compilation stage, pass -S to

option in gcc:gcc -S Hello.c

This will create a Hello.s file.

If you look at the “Hello.s” file you will find all the assembly command.

Assembly

- It convert the assembly instruction to object code.
- It contain the actual instruction to be run by the

target processorTo see assembly stage, pass -c to option in

gcc:

gcc -c Hello.c

This will create a Hello.o file.

"Hello.o" is a Machine Level Languge file which cannot be read by the text editor. To read the file runfollowing command in the terminal.

objdump -D Hello.o

This will shows the object file in terminal.

Linking

- Object code generated by the assembly stage is composed of

  machine instruction that the processor understands but they are not

  arrange in an order.

- To produce an executable program existing pieces have to be rearranged and missing ones arefulfilled.

To see Linking stage, pass -o to

option in gcc:gcc -o Hello.c

This will create Hello.exe which is an executable file.

To see all the files at one command use: gcc -o Hello–save-temps Hello.c

6. FURTHER IMPROVEMENT

This game is developed by a group of four students who are currently studying in BCT 1st Year and is purely for study purpose. Developed with the intention to excel our C Programming knowledge, the program might contain few error which may cause the game to be misguided.

From what seems to us, the game is yet to be developed for single player mode which would let the user compete with the Computer. This required more than basic knowledge of C Programming and due to limited time, we couldn't develop this feature. Hopefully, it will be given green light in days to come in our Version 2.0
Secondly, the graphics enhancement of the game is also stagnant because of our little knowledge in this field. Sound effects during selection of options was major point of concern which again could not be accomplished due to limited time and manpower.

We would also like to let you know that this game is still under development mode and you are the inspector of this. Please do let us know your thoughts on this and contact the developers with the areas of improvement that enhances gaming experience.

Developers:

1. Neeka Maharjan
2. Punam Adhikari
3. Sudip Tiwari
4. Susheel Thapa

Conclusion:

The sole purpose of this project is to make us students dive into real life scenarios where we can utilize and polish our programming skills in order to solve them using both our analytic and programming skills. The programming language used here, by our experience throughout this first semester, was C Programming. Among the various Integrated Development Environment (IDEs) and compilers available, Visual Studio Code with Mingw-w64 was utilized.

C Programming is a High Level Language that uses human language-like syntax to code and is easy to understand and debug. Through basic programming knowledge of C, including Control Statements, Looping, Functions, File Handling, Strings, Arrays and so on and so forth, this level has been reached, whereby a group comprised of four BCT First Semester students could develop a Tic Tac Toe game program. Speaking of work division among four members of the group, the concept of function has helped us a lot. The program was divided into major 4 functions and each function was coded by one of the group members. Even after final compilation of the various functions, the program was tested and debugged by our group members and some evaluation team members.

Although all of our heart and soul was gushed into coding this game, it is well known that any program leaves some room for improvement. This program, too, expects certain of the conditions to be met by the users and at some times, limits users from customizing it. The effort, however, was always to try our best to make the program user friendly and explore our programming knowledge in doing so. The deviation in our goal to be met might be due to the fact that the content in our C Programming course was limited and there was not enough time for us to research. Nevertheless, this is not an excuse for us to take a setback but to explore more and develop this into a more commercial one. This Project is just the beginning of formal real life problem solving experience for many of us and we hope, through similar support from our facilitators and department, we would only grow higher.