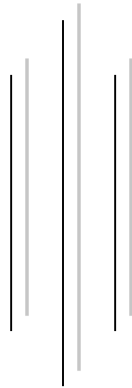




**TRIBHUWAN UNIVERSITY**  
**INSTITUTE OF ENGINEERING**  
**Pulchowk Campus, Lalitpur**

**Report on the mini project work of computer  
on the topic  
“SNAKE”**



**SUBMITTED BY:**

- Prashant Bhatta (073BEX425)
- Rishav Mani Sharma (073BEX433)
- Sagar Bhandari (073BEX435)
- Ram Sharan Rimal (073BEX431)

**SUBMITTED TO:**

Department of Electronics and Computer

**ACKNOWLEDGEMENT**

It would have been very hard for us to complete our project on time and in efficient way. Without the support, guidance and suggestion from our faculty members and friends, we could not have completed this project on time and in efficient way.

First of all, we have no words that can adequately carry our sincere gratitude to, Mr. Sanjay Bhandari, *Subject teacher* for providing us the all the opportunity to conduct the project. We are very grateful to him for his valuable suggestions and encouragement for this project work. He has provided us all sort of basic needs and techniques essential for carrying out this project work.

We would also like to acknowledge and thank all the faculty members for their kind support and guidance as well as their inspiration to move forward for the completion of project on time. We can't leave to thank our dear friends as they have provided us with their valuable comments and suggestions for the completion.

We would also like to thank all the writer of the books that we have used. Without their books, it would have been hard and tedious for us to get knowledge and information regarding the subject matter and it might have taken long time to complete this project.

Finally, we would like to provide our sincere gratitude to Anmol Poudel who have helped us in completing this project in highly efficient way.

Thanks.

**TABLE OF CONTENTS**

<b><u>Topics</u></b>	<b><u>Page No.</u></b>
<b>1. Background.....</b>	<b>4</b>
<b>2. Introduction.....</b>	<b>6</b>
<b>3. Flowchart.....</b>	<b>7</b>
<b>4. Working principle .....</b>	<b>8</b>
<b>5. Source Code.....</b>	<b>8</b>
<b>6. Output and results.....</b>	<b>18</b>
<b>7. Discussion.....</b>	<b>23</b>
<b>8. References.....</b>	<b>23</b>

## **BACKGROUND**

Present world which dependent on the accelerating technology is in ay bounded to our lifestyle that no one imagine parting from it. While programming is an overlooked part of today's' intricate gadgets, it is a key part nonetheless .The world is extremely reliant on its programmers. Without them, the world as we know it would be forced to say good bye to many luxuries. The "global village" concept, which refers to the connection of the world as a result of telecommunication, would be eradicated concept. The electronics that we use in our lives would no longer exist. And , surprisingly ,many of the common objects or conditions we see every day ,whether visibly electronics or not ,would cease to work .These include bridges, space explorations ,and even some medicines, which are all based on some level of programming.

Before this we have started a small effort to enter this fascinating virtual world. On the way of our C-Programming projects ,which is in fact an effective way for letting students discover the world of coding and develop the teamwork .

C is a programming language developed at AT & T's Bell laboratories of USA in 1972. It was designed and written by Denis Ritchie.

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. 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).

Communicating with a computer involves speaking the language the computer understands, which immediately rules out English as the language of communication with computer. However, there is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets or characters used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences are combined to form paragraphs. Learning C is similar and much easier.

Therefore, instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how the constants, variables and keywords are constructed, and finally how these are combined to form an instruction. A group of instructions would be combined later on to form a program.

**Features of good program:-**

Programmers must keep in mind the following things before writing a program in order to have a good program.

★ **Integrity:-**

The calculations used in the program should be very accurate. It must provide the desired output (functionally correct) for the given input. It must do the work according to the specification.

★ **Clarity:-**

The program should be well readable to aid maintenance later. This can be provided inline-documentation or external documentation. On inline documentation, the function of each piece of code is defined within the program itself. In the external documentation, a separate report includes the working principle of each module inside the program.

★ **Simplicity:-**

The program should be able to express the logic in a considerably simple way. This feature enhances Integrity and Clarity. Same problem can be done in two or more ways, but one needs to choose the simplest way to solve the problem.

★ **Efficiency:-**

The program should have a good compromise between Time and space used, i.e. it should run as fast as possible (time) with the minimum memory requirement (space).

★ **Modularity:-**

Program should be separated to different logical and self-contained modules. If the entire program is divided into simpler contained modules then one can easily understand what's happening inside it.

★ **Generality:-**

The program should be as general as possible within certain limits.

★ **Robustness:-**

Program must be fault-tolerant as much as possible. A program cannot be 100% full-proof, so it must be built in such a way that, if some unavoidable circumstance appears, then it tackles with this without being crashed.

★ **Security:-**

A program must be secured enough so as to avoid tampering from unwanted people. All the loopholes in the programs must be avoided as much as possible.

## **INTRODUCTION**

“Snake” is the program written on C language using Turbo c++ compiler. We have tried to replicate the snake game we used to play in nokia .We have tried our best to make it same like it but we have not included more level and complexity as it will make our project long .It is a single player game where one can control a snake using four direction keys. In our program we have used 'a' , 's', 'd' , 'w' as direction keys . A player can score by eating food provided without touching the side walls or the obstacles present .It can be very entertaining and good to play in leisure time.

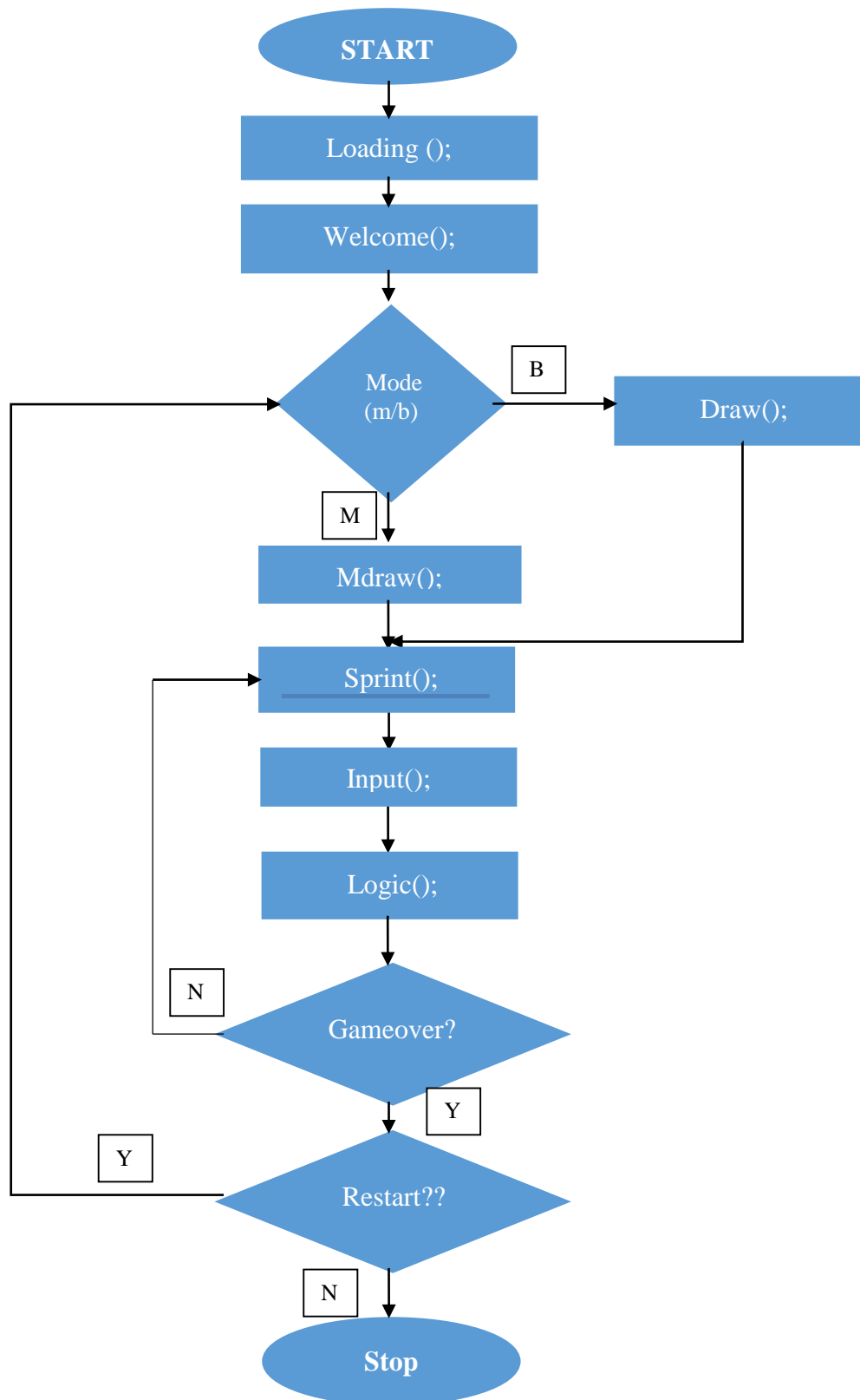
It has been made very user friendly and we have hoped that the user may feel very comfortable to use this game without having any such difficulty. It has two mode of playing . One can either play box mode or maze mode. Box mode features platform surrounded by walls and maze mode along with side walls features some more obstacles for more fun .Other than game mode we have also included 5 levels in this game .Levels indicate the speed of the snake's movement .

Before starting the development of this program, we had certain objectives in mind to fulfill which are listed as below:

- To use different in-built functions of C and also make the user defined functions for making the program look simple.
- To make the use of simple code as far as possible to make the program look simple and easier to understand.
- To make the game more user friendly.

## **OBJECTIVE**

- To learn about C-programming via a project completely based on it.
- To extract about structure, array, pointers with their uses in the project.
- To impart knowledge about simple game developing with C.
- To develop skills regarding group work and presentation.
- To know about the use of different datatypes, functions and header files in C.
- To understand the features of C programming and its importance.

**FLOWCHART:-**

### WORKING PRINCIPLE:

At first the loading (); calls the loading process of our program and then welcome(); is executed. Then the necessary data required for game are taken from user where player can choose gaming mode and level of game. Here in this game level indicates the speed of snake. And then according to the mode selected by user the background screen is executed. And then the main program is executed which do have a infinity loop containing input (); logic(); and sprint();(snake print functions which executes until game over. The criteria for game to be over are either snake touches its own tail or it hits the walls and obstacles. The above flowchart shows complete process of execution of our program.

### SOURCE CODE:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<time.h>
int highscore=300;
const int width=75;
const int height=20;
char maze,pp=0,name[20];
int i,j,k,d=219,cheat;
int gameover,print=0,level,p;
int
linex[100]={15,15,15,15,15,15,15,15,15,15,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
,20,21,22,23,24,60,60,60,60,60,60,60,60,60,60,60,53,54,55,56,57,58,59,60,61,62,63,64,65,66,6
7,68,69,70,71,72,73,74,75,76,77,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,
47,38,38,38,38,38,38,38};
int
liney[100]={0,1,2,3,4,5,6,7,8,9,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14
,14,14,14,14,14,0,1,2,3,4,5,6,7,8,9,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,1
4,14,14,14,14,14,14,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,13,14,15,16,17,18,19};
int x,y,fruitx,fruity,score;
int tailx[100],taily[100];
int ntail=0;
int dir,xxx,yyy,flagg;
union REGS ii,oo;
void callmouse()
{
    ii.x.ax=1;
    int86(0x33,&ii,&oo);
}
int clickk()
{
```



```
int click;
ii.x.ax = 3;
int86(0x33,&ii,&oo);
click=oo.x.bx;
return click;
}

void mousehide()
{
    ii.x.ax=2;
    int86(0x33,&ii,&oo);
}

void sett(int *a, int *b)
{
    ii.x.ax=3;
    *a=oo.x.cx;
    *b=oo.x.dx;
    int86(0x33,&ii,&oo);
}

void mouse()
{
    int a,b,click;
    clrscr();
    callmouse();
    gotoxy(30,9);
    printf("!! SNAKE GAME !!");
    printf("\n\n\t\t\t START");
    printf("\n\t\t\t EXIT");
do
    {
        click=clickk();
        sett(&a,&b);
        gotoxy(10,3);
        printf("position : (%d,%d)\n",a,b);
        printf("      click:%d",click);
        if (a>=264&&a<=303&&b>=80&&b<=86&&click==1)
            break;
        else if(a>=264&&a<=294&&b>=87&&b<=93&&click==1)
            {
                gameover=1;
                break;
            }
        else
            continue;
    }while(!kbhit());
getch();
mousehide();
printf("exit");
```

```

getch();
}
void setup()
{
    gameover = 0;
    dir = 0;
    fruitx=rand()%width;
    fruity=rand()%height;
    x=width/2-8;
    y=height/2;
    score = 0;
}
void loading()
{
    int a=177,b=219,i,j;
    clrscr();
    gotoxy(30,11);
    for(j=0;j<=20;j++)
    {
        printf("%c",a);
    }
    gotoxy(30,11);
    for(i=0;i<=20;i++)
    {
        gotoxy(25,10);
        printf("    LOADING  %d%",(i*5));
        gotoxy(30+i,11);
        printf("%c",b);
        sound(1000);
        delay(50);
        nosound();
        delay(50);
    }
    getch();
    sound(1000);
    delay(50);
    nosound();
}
welcome()
{
    char a[17]="WELCOME TO THE";
    char b[15]="C-PROGRAMMING";
    char c[15]="SNAKE GAME";
    gotoxy(33,10);
    for (i=0;i<=14;i++)
    {
        if(kbhit())

```

```

        break;
        sound(1000);
        printf("%c",a[i]);
        delay(50);
        nosound();
        delay(100);
    }
    printf("\n\t\t\t");
    for (i=0;i<=14;i++)
    {
        if(kbhit())
            break;
        sound(1000);
        printf("%c",b[i]);
        delay(50);
        nosound();
        delay(100);
    }
    printf("\n\t\t\t");
    for (i=0;i<=10;i++)
    {
        if(kbhit())
            break;
        sound(1000);
        printf("%c",c[i]);
        delay(50);
        nosound();
        delay(100);
    }
    getch();
    fflush(stdin);
    sound(1000);
    delay(100);
    nosound();
    return 0;
}

void input()
{
    if(kbhit())
    {
        dir=0;
        switch (getch())
        {
            case 'a':
                dir = 1;
                break;

```

```

        case 'd':
            dir = 2;
            break;
        case 'w':
            dir = 3;
            break;
        case 's':
            dir = 4;
            break;
        case 'x':
            gameover=1;
            break;
    }
    fflush(stdin);
}

}
void logic()
{
    int px=tailx[0],py=taily[0],p2x,p2y;
    tailx[0]=x;
    taily[0]=y;
    for(i=1;i<=ntail;i++)
    {
        if ((x==tailx[i])&&(y==taily[i]))
        {
            gameover=1;
        }
        p2x=tailx[i];
        p2y=taily[i];
        tailx[i]=px;
        taily[i]=py;
        px=p2x;
        py=p2y;
    }
    switch(dir)
    {
        case 1:
            {
                x--;
            }
            break;
        case 2:
            {
                x++;
            }
            break;
    }
}

```

```

case 3:
{
y--;
}
break;
case 4:
{
y++;
}
break;
}
if ((x<=0) || (x>=width+1) || (y<=1) || (y>=height+2))
{
gameover = 1;
ntail=0;
}
if((maze=='m') || (maze=='M'))
{
for(p=0;p<100;p++)
{
if(x==linex[p]+1&&y==liney[p]+2)
gameover=1;
}
}
if ((x==fruitx)&&(y==fruity))
{
ntail=ntail+1;
score+=10;
if(score>highscore)
highscore=score;
do
{
flagg=0;
fruitx=rand()%72+2;
fruity=rand()%18+2;

for(j=0;j<100;j++)
{
if(((fruitx==linex[j])&&(fruity==liney[j])) || fruitx==0 || fruity==0)
{
flagg=1;
break;
}
}
}while (flagg==0);
sound(200);

```

```

        delay(100);
        nosound();
    }
}
void draw()
{
    clrscr();
    for (i=0;i<width+2;i++)
    {
        printf("%c",d);
    }
    for (i=0;i<height;i++)
    {
        for (j=0;j<=width;j++)
        {
            pp=0;
            if(j==0)
            {
                printf("\n%c",d);
            }
            else
            {
                print=0;

                if (print==0)
                {
                    printf(" ");
                }
            }
            if (j==width)
            {
                printf("%c",d);
            }
        }
    }
    printf("\n");
    for (i=0;i<width+2;i++)
        printf("%c",d);
    printf("\n\nScore = %d \t\t\tHighscore = %d",score,highscore);
}
void mdraw()
{
    clrscr();
    for (i=0;i<width+2;i++)
    {

```

Press X to quit the

```

        printf("%c",d);
    }
    for (i=0;i<height;i++)
    {
        for (j=0;j<=width;j++)
        {
            pp=0;
            if(j==0)
            {
                printf("\n%c",d);
            }
            else
            {
                print=0;

                for(p=0;p<100;p++)
                {
                    if(((i==liney[p])&&(j==linex[p]))&&pp!=1)
                    {
                        printf("%c",d);
                        print=1;
                    }
                    if((x==linex[p]&&(y==liney[p])))
                    {
                        gameover=1;
                    }
                }
                if (print==0)
                {
                    printf(" ");
                }
            }
            if (j==width)
            {
                printf("%c",d);
            }
        }
    }
    printf("\n");
    for (i=0;i<width+2;i++)
        printf("%c",d);
    printf("\n\nScore = %d \t\t\tHighscore = %d",score,highscore);
}
void sprint()
{

```

Press X to quit the

```

gotoxy(xxx,yyy);
printf(" ");
gotoxy(x,y);
xxx=x,yyy=y;
printf("%c",2);
if (dir!=0)
{
    sound(2000);
    delay(10);
    nosound();
}
gotoxy(fruitx,fruity);
printf("%c",4);
for(k=0;k<ntail;k++)
{
    gotoxy(tailx[k],taily[k]);
    printf("o");
    gotoxy(tailx[ntail],taily[ntail]);
    printf(" ");
}
gotoxy(1,24);
printf("Score = %d \t\t\tHighscore = %d    Press X to quit the game",score,highscore);
switch(level)
{
    case '1':
        delay(120);
    case '2':
        delay(100);
    case '3':
        delay(80);
    case '4':
        delay(60);
    case '5':
        delay(40);
}
}
void main()
{
    mouse();
    loading();
    clrscr();
    gotoxy(25,11);
    printf("Please enter your name :-");
    gets(name);
    fflush(stdin);
    clrscr();
    welcome();

```



```

    clrscr();
    top:
    gotoxy(25,10);
    printf("\n\t\t If you want to play in maze, press m\n\t\t If you want to play in box,
press b :- ");
    maze=getch();
    sound(1000);
    clrscr();
    gotoxy(25,10);
    printf("\n\t\t\t Enter level from 1 to 5 \n\t\t\t\t :- ");
    delay(100);
    nosound();
    level=getch();
    fflush(stdin);
    sound(1000);
    delay(100);
    nosound();
    setup();
    clrscr();
    if((maze=='m') || (maze=='M'))
        mdraw();
    else
        draw();
    while(gameover!=1)
    {
        rand();
        sprint();
        input();
        logic();
    }
    sound(100);
    delay(1000);
    nosound();
    getch();
    clrscr();
    gotoxy(30,10);
    printf("\tGAME OVER\n\t\t\t\t%s Your score is = %d",name,score);
    if(score==highscore)
        printf("\n\t\tCongratulations!! You beat the highscore");
    else if((score+20==highscore) || (score+10==highscore))
        printf("\n\t\t Oops!! You nearly beat the highscore");
    else
        printf("\n\t\t You couldn't beat the high score = %d",highscore);
    getch();
    sound(1000);
    clrscr();
    gotoxy(20,10);

```

```
printf("\n\t\tDo you want to continue? Press y\n\t\tOtherwise press any key :- ");
delay(100);
nosound();
switch(getch())
{
    case 'y':
    {
        gameover=0;
        ntail=0;
        sound(1000);
        delay(100);
        nosound();
        goto top;
    }
    default:
        sound(1000);
        delay(100);
        nosound();
}
}
```

### Output:-

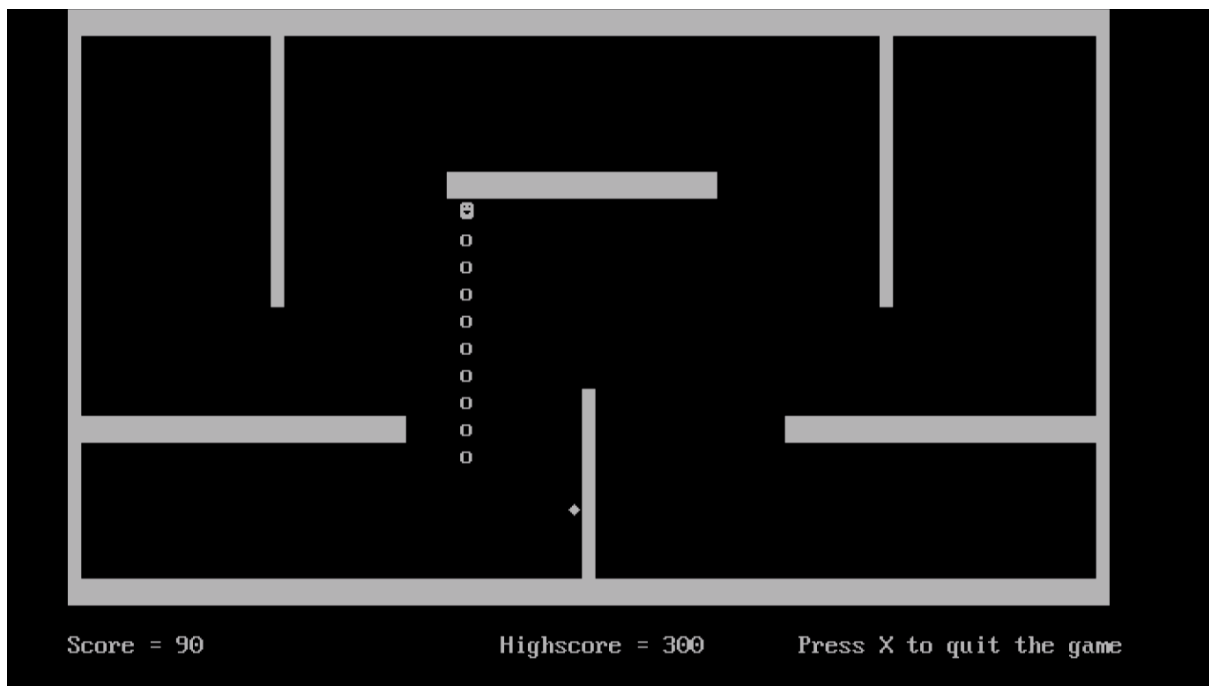


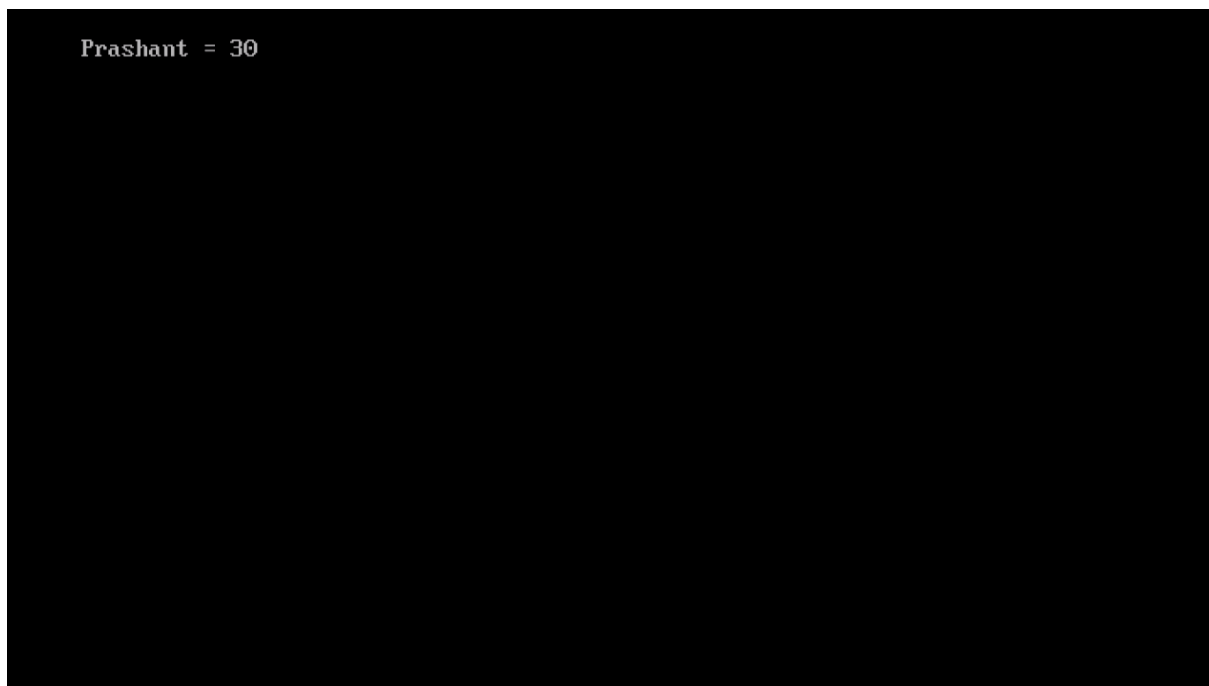
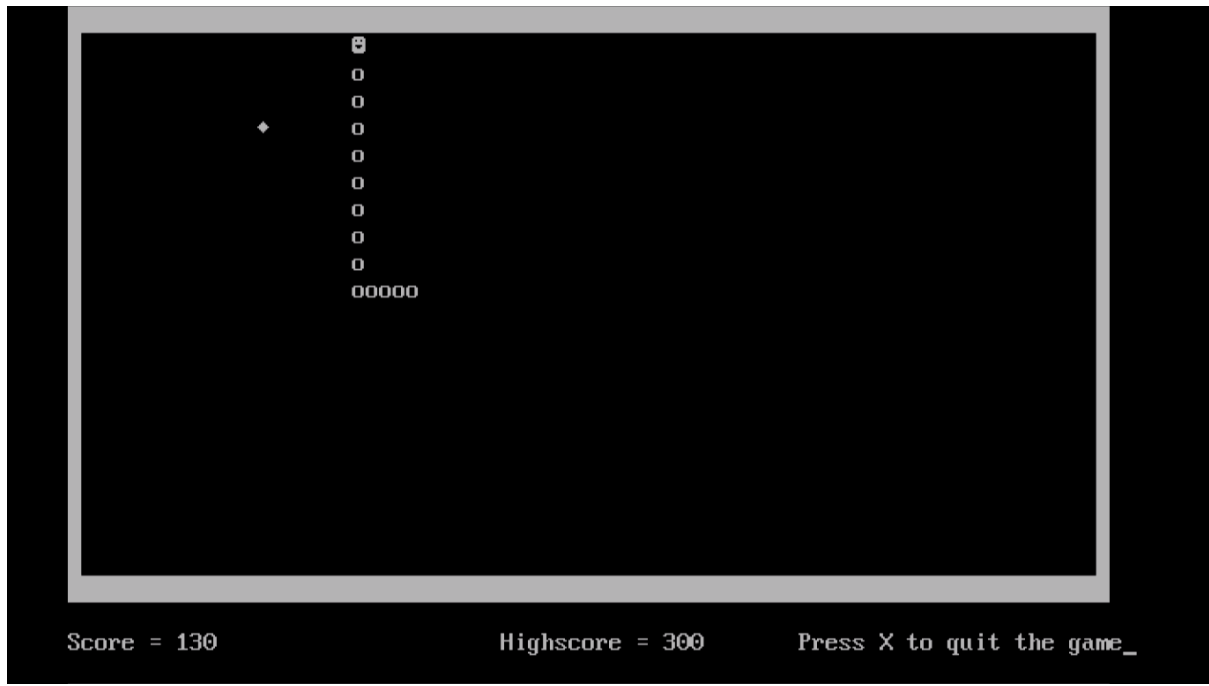
Please enter your name :-Prashant

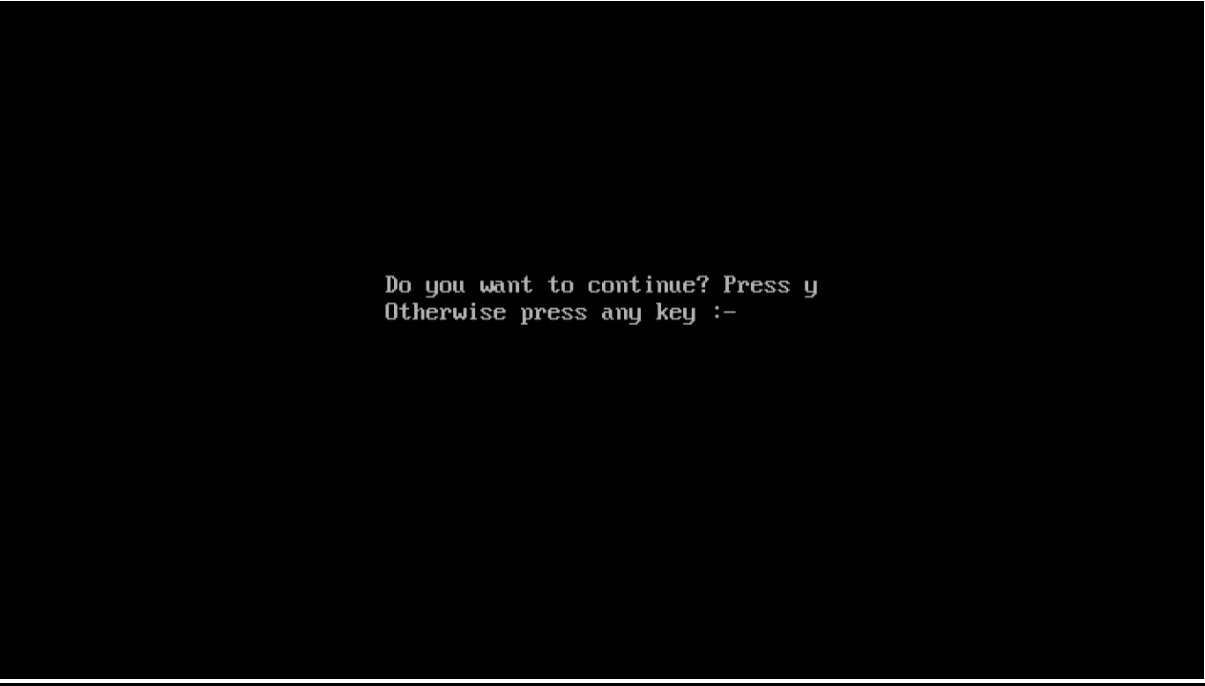
WELCOME TO THE  
C-PROGRAMMING  
SNAKE GAME

If you want to play in maze, press m  
If you want to play in box, press b :-

Enter level from 1 to 5  
:-







```
Do you want to continue? Press y  
Otherwise press any key :-
```

## **DISCUSSION**

During the completion of our project we faced lot of problems .Among them the flickering problem was our major problem .We tackled it by keeping our background screen constant and only replacing the moving parts. On the other hand due to lack of time and tedious nature of that work we weren't able to develop a combo of snake and pacman which would include ghosts and obstacles too. If this happened the game would have been more challenging and interesting to play.

## **REFERENCES**

- Programming with C by Bryon S.Gottfried
- ANSI C, by Balaguruswamy
- Turbo C++ v 3.0 Help
- Borland C++ v 5.03 Help
- Internet