Snake game using C Data Structures

By:

16bce0659 -A Bhanu Datta

16bce0808 -Shubham

16bce2064 -Vishnu Vardhan Reddy

16bei0036 -Anshul

Faculty:Chandramohan

Session overview

- **Topic**
- **Objective**
- **Introduction**
- **Previous work**
- **Proposed work**
- **Result**
- **Performance analysis**
- **references**

Aim:

- To bring the fun and simplicity of snake game with some new features.
- Explores a new dimension in the traditional snake game to make it more interesting and challenging.

- In this game player moves the snake in a window and makes it to eat the food to score points and increase the length of the snake.
- Objective: Console application without graphic library.
- Planning to use many user defined functions in it.
- Going to use the following data structures and commands: linked list or array ,typedef struct , gotoxy function.

We are not going to use every data structure we mentioned .We might change it according to the algorithm and code.

Uses of this project

- It is going to be a fun project and it will be very easy in controlling for the user who plays the game .
- We are planning to add features like high score and the user will be given instructions regarding how to play the game .
- We have played many snake games in mobiles since our childhood but creating our own game is always special.
- Explanation of the data structures which we are going to use in this project is given in the next slide.

Definition

- **Array of structures**: C structure is a collection of different data types which are grouped together whereas array of structures is a collection of structures. This is also called as structure array in c.
- Arrays are helpful largely as a result of the element indices will be computed at run time. Among different things, this feature permits one iterative statement to process arbitrarily several components of an array.
- Array types are usually implemented by array structures; but, in some languages they will be enforced by hash tables, joined lists, search trees, or different knowledge structures.
- **Typedef struct**: you can use typedef to give a name to your user defined data types as well and then use that data type to define structure variables. When trying to build a basic linked list we can use typedef struct.

- Typedefs is used both to supply additional clarity to your code and to form it easier to form changes to the underlying data types that you just use.
- One purpose of getting types is to make sure that variables are continuously employed in the method that they were supposed to be used.
- As such, it is often accustomed change the syntax of declaring advanced knowledge structures consisting of struct and union varieties, but is simply as common in providing specific descriptive kind names for whole number knowledge varieties of varied lengths.
- **Linked list** :It is a sequence of information structures that area unit connected along through links every link contains a association to a different link.
- In computer science, a linked list is a linear collection of data elements, called nodes, each inform to the next node by means that of a pointer.
- It is a data structure consisting of a bunch of nodes which together represent a sequence. Under the simplest kind, each node is composed of information and a reference (in different words, a link) to the next node within the sequence.
- This structure allows for economical insertion or removal of components from any position in the sequence throughout iteration. More advanced variants add extra links, allowing economical insertion or removal from discretionary part references.
- **Gotoxy function:** It is a function or procedure that positions the cursor at (X,Y) , X in horizontal, Y in vertical direction relative to the origin of the current window .
- It is used to relocate the cursor on your console or terminal at a particular co ordinate according to parameters passed to it.
- The gotoxy() funtion is used to simply move the cursor on your monitor screen wherever desired.
- It is declared in the "conio.h" header file.
- It takes 2 parameters, gotoxy(int x, int y). The 1st parameter specifies the x-co-ordinate on the screen, i.e., horizontal position(from left to right) on the screen. The 2nd parameter specifies the y-co-ordinate on the screen, i.e., Vertical position(from top to bottom) on the screen. Thus, gotoxy(1,1) tells the compiler to move the cursor to the left-top-most corner of the screen.
- Linked lists are among the simplest and commonest information structures. They can be accustomed implement many alternative common abstract information varieties, including lists (the abstract data type), stacks, queues, associative arrays, and S-expressions.

- Though it is not uncommon to implement the opposite information structures directly while not employing a list because the basis of implementation.
- The principal advantage of a linked list over a standard array is that the list parts will simply be inserted or removed while not reallocation or reorganization of the whole structure as a result of the info things needn't be keep contiguously in memory or on disk, whereas associate degree array needs to be declared within the ASCII text file, before assembling and running the program.
- Linked lists permit insertion and removal of nodes at any purpose within the list, and might do thus with a continuing variety of operations if the link previous to the link being accessorial or removed is maintained throughout list traversal.

<u>Previous work by others</u>

- If you see other snake games very few have the option of checking high scores and previous scores by date and time.
- This a simple game we did not want to make it complex for the player. Even the code is easy to understand.
- Others might have included graphics but we did not include it.

<u>Project Plan</u>

- Planning to divide the work almost equally to everyone in the group.
- Planning to use user defined functions for:

❑ Food of the snake

❑ Score in the game

❑ Bending in the turns

❑ High score to compare with others

❑ Length of snake-it increases when it eats food

❑ Movement of snake and loading of the game

❑ Speed of the snake in different directions

❑ Exiting the game

- To give a warm welcome to the player and also some basic instructions to the game.
- Three lives will be given to players to prove their game standards

- Option to pause the game at any point and option to exit the game at any moment.
- For movement we are using *Gotoxy* function which places the cursor in places we require.
- Not including graphics, just want to make a simple snake game using data structures.
- Taking help from the faculty or seniors if wanted in some areas we lag clarity.

Implementation Details

- For the movement of snake, for moving up, down, left or right we increase or decrease the x or y co ordinate accordingly and call the function(i.e.direction-up or down or…..).
- At the end we define every function(direction) in detail so that get implemented accordingly.
- If the x and y co ordinate of the food and head of snake is same it is counted as a point and the food disappears and is placed somewhere else, length also increases.
- At the start, length of snake is 5 so score is equal to length of snake – 5.
- Printing the loading in the middle of the screen by gotoxy function.
- Leaving the head, the body of the snake is 4 because it needs 4 elements to touch its own body.
- If the co ordinates of the head of the snake and any part of its body is equal then life decreases.
- If lives become zero then the game exits and a notice saying the lives are over better luck next time and an option to quit the game appears.
- After playing the game, we call the score and ask to enter your name. Using upper we change the first letter after space to capital.
- Giving a chance to see the previous record if pressed the given character. If the character is same then we call the record, all the previous games score appears.

Diagram for user defined functions used in the game

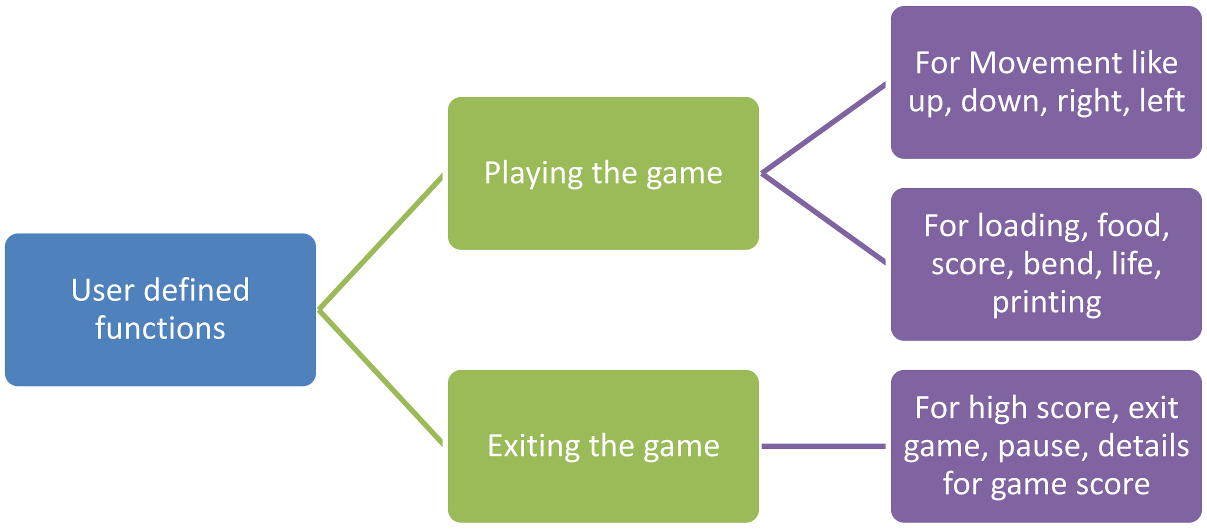
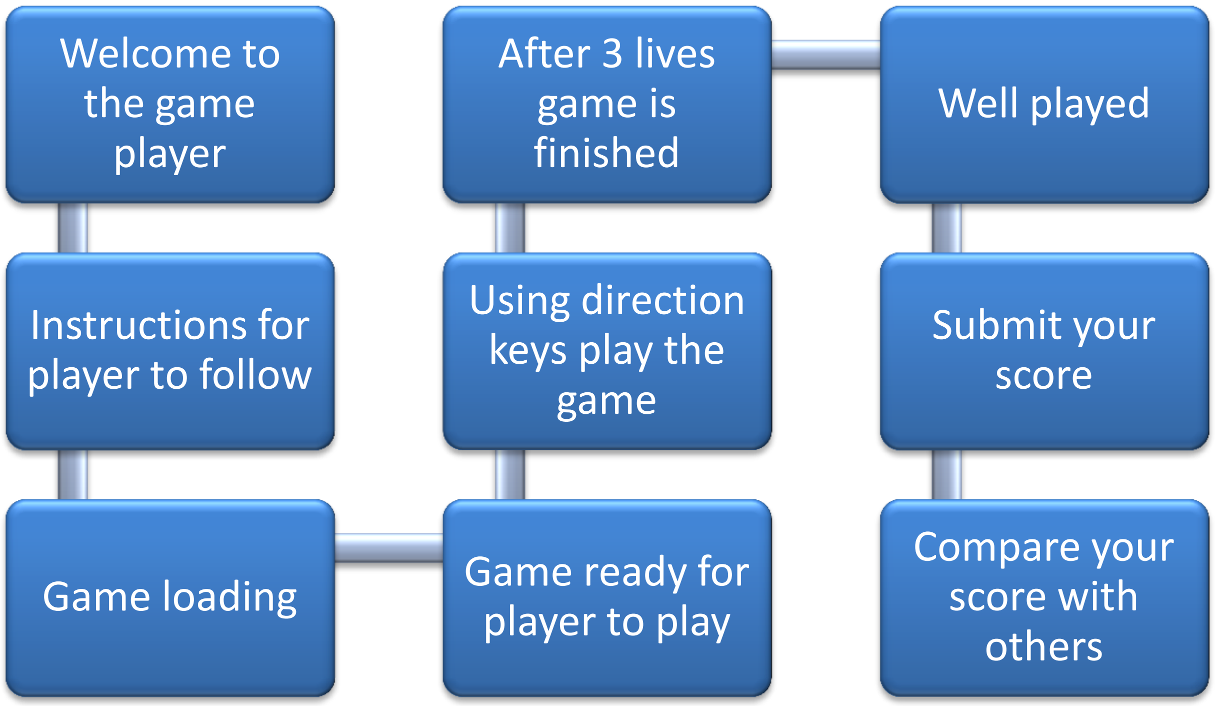
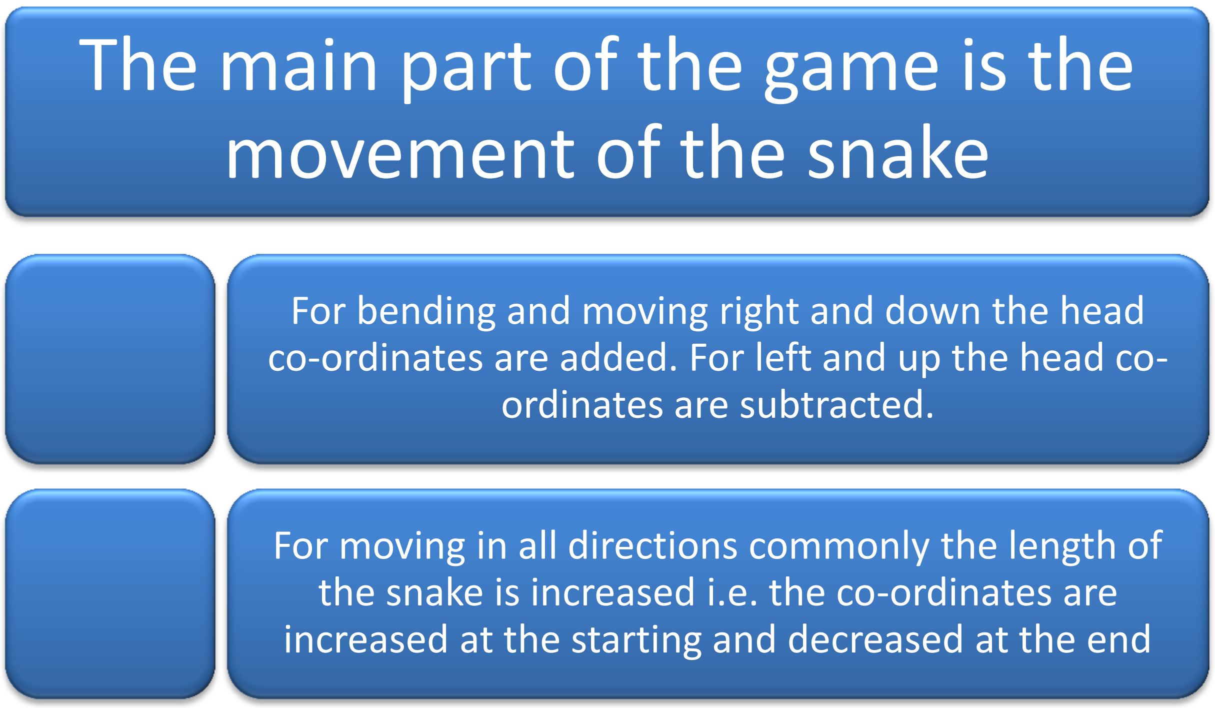

User defined functions
Playing the game
Exiting the game
For Movement like up, down, right, left
For loading, food, score, bend, life, printing
For high score, exit game, pause, details for game score

Diagram for playing the game


Welcome to the game player
Instructions for player to follow
Game loading
Game ready for player to play
Using direction keys play the game
After 3 lives game is finished
Well played
Submit your score
Compare your score with others

<u>Diagram for implementation of movement of snake in the game</u>

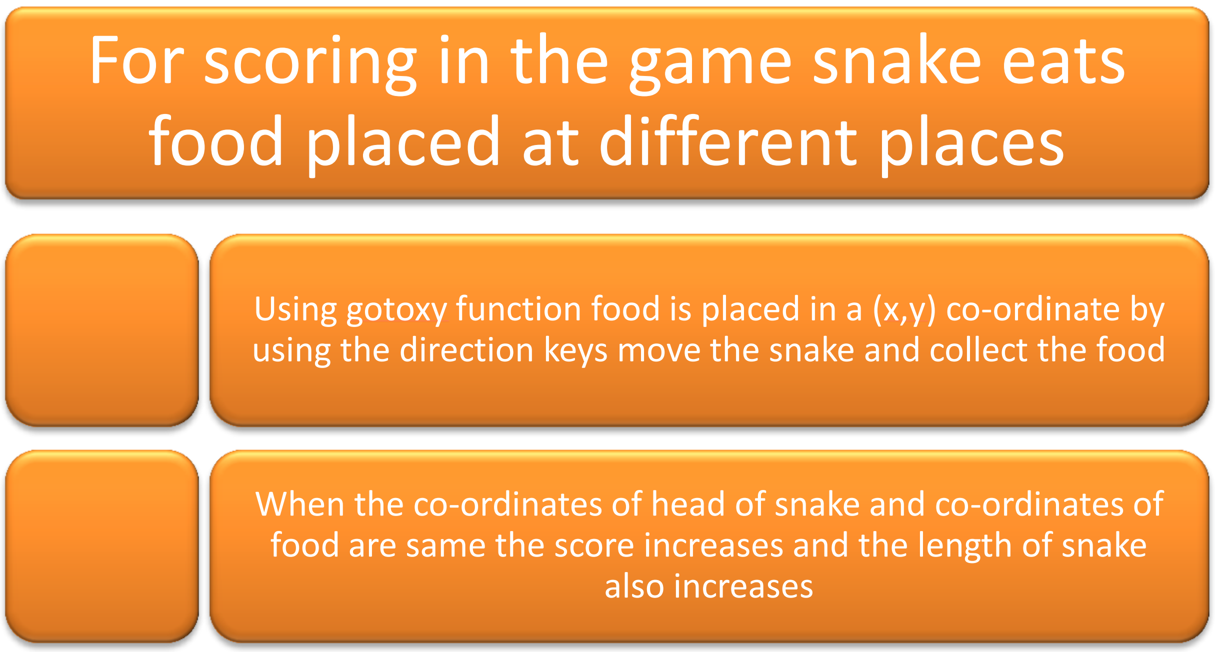


The main part of the game is the movement of the snake

For bending and moving right and down the head co-ordinates are added. For left and up the head co-ordinates are subtracted.

For moving in all directions commonly the length of the snake is increased i.e. the co-ordinates are increased at the starting and decreased at the end

<u>Diagram for implementation of food and score of snake game</u>



For scoring in the game snake eats food placed at different places

Using gotoxy function food is placed in a (x,y) co-ordinate by using the direction keys move the snake and collect the food

When the co-ordinates of head of snake and co-ordinates of food are same the score increases and the length of snake also increases

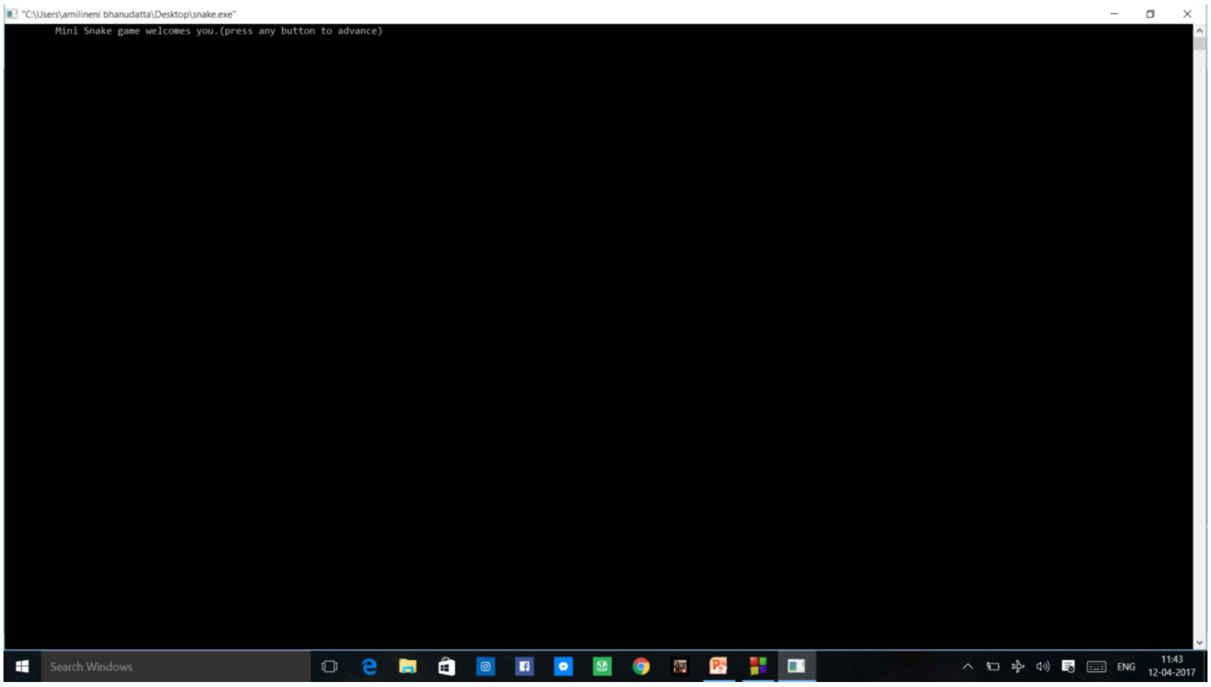
Diagram for implementation of death in snake game

For losing a life in the game the snake has to come across itself or should hit a wall

If the co-ordinates of the head of the snake are equal in any case to the co-ordinates of body then life is decreased

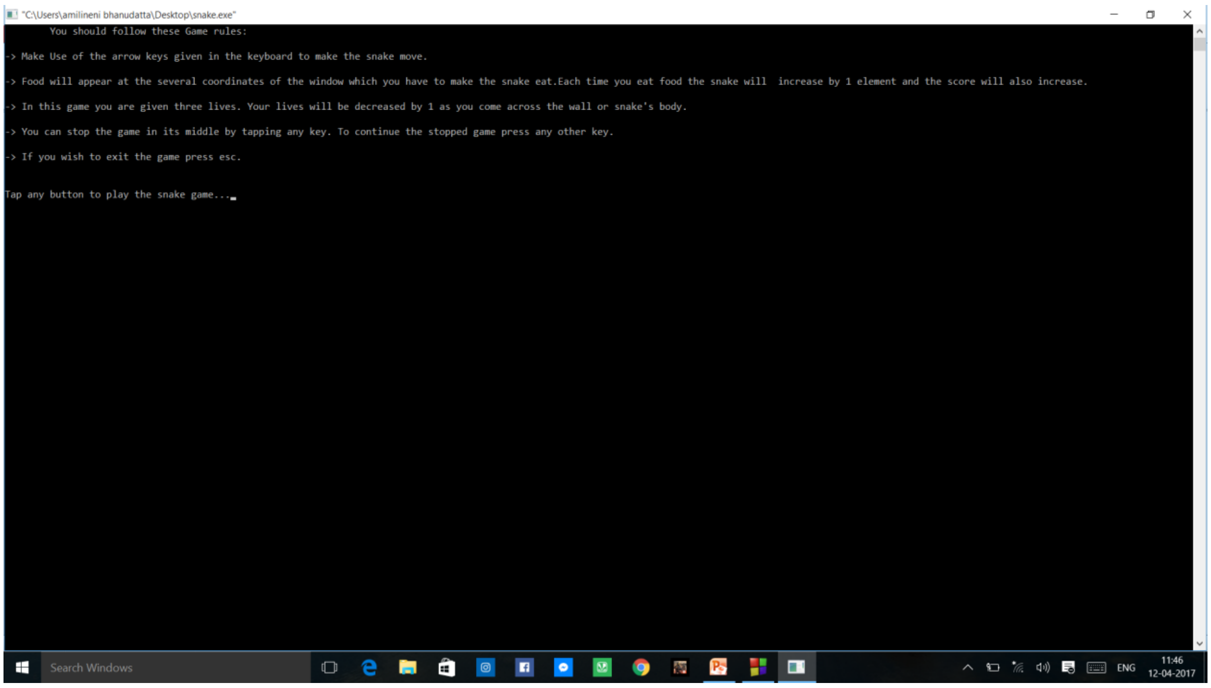
If the co-ordinates of the head of the snake are equal in any case to the co-ordinates of the surrounding walls then life is decreased

Welcome to the mini snake game



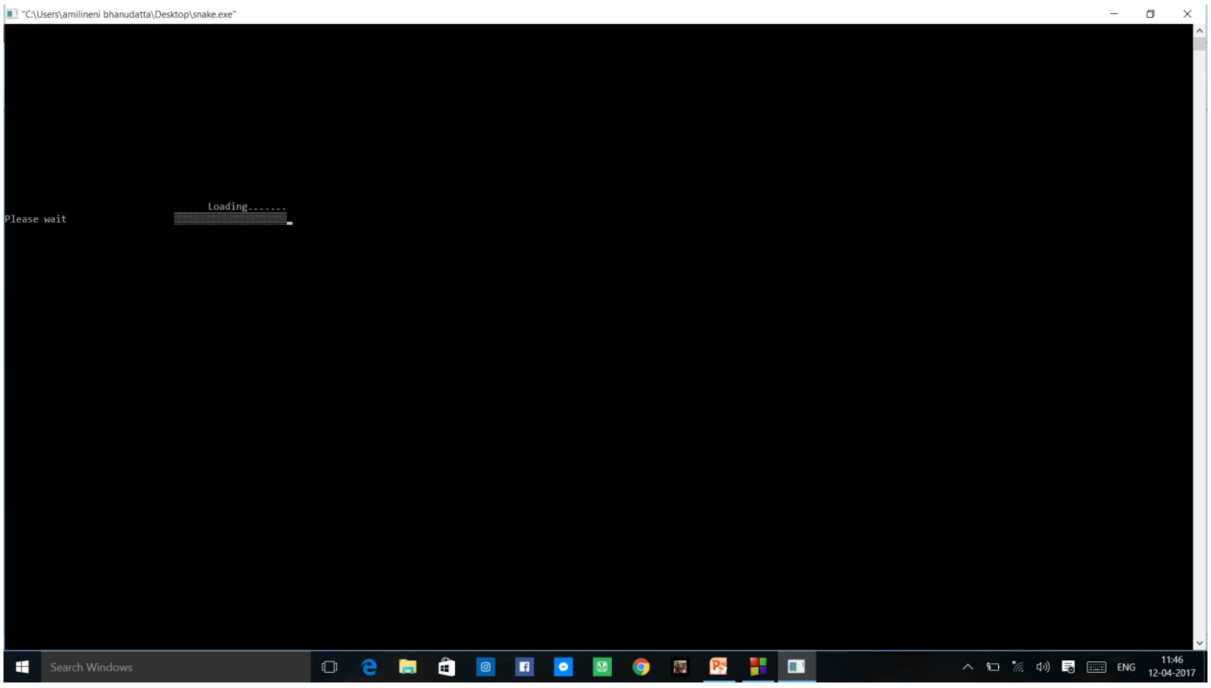
```
       Mini Snake game welcomes you.(press any button to advance)
```

Read the instructions and proceed


"C:\Users\amilineni bhanudatta\Desktop\snake.exe"

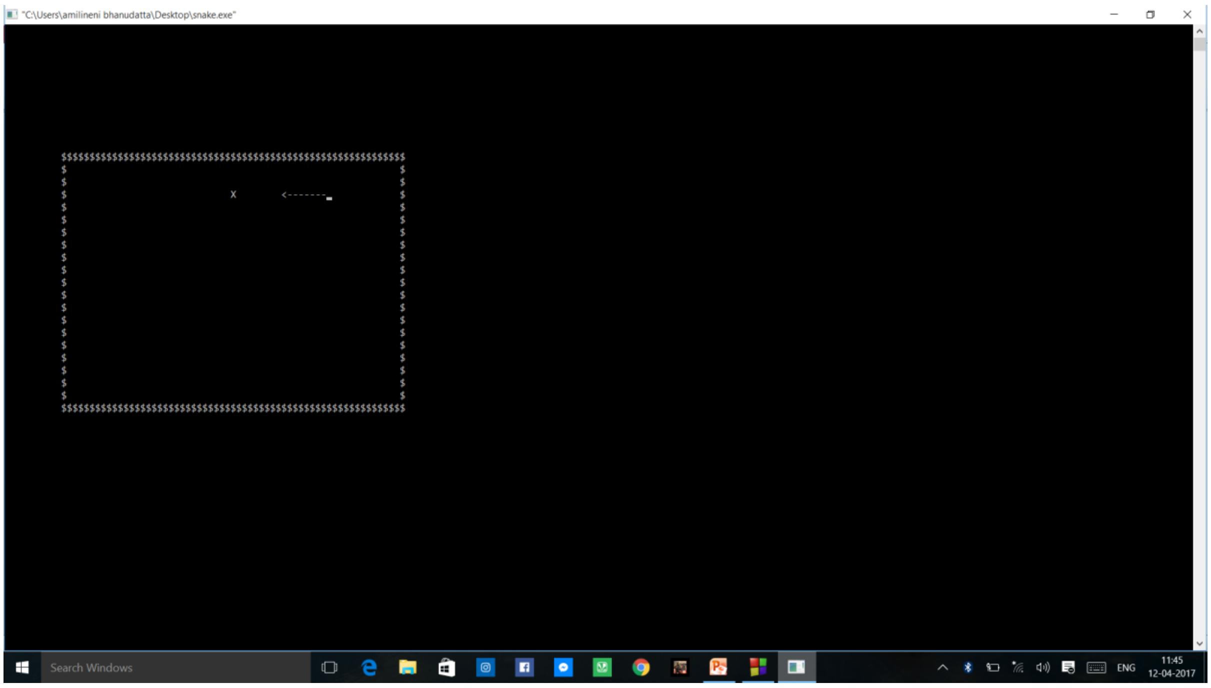
```
       You should follow these Game rules:

-> Make Use of the arrow keys given in the keyboard to make the snake move.

-> Food will appear at the several coordinates of the window which you have to make the snake eat.Each time you eat food the snake will  increase by 1 element and the score will also increase.

-> In this game you are given three lives. Your lives will be decreased by 1 as you come across the wall or snake's body.

-> You can stop the game in its middle by tapping any key. To continue the stopped game press any other key.

-> If you wish to exit the game press esc.


Tap any button to play the snake game...
```

Wait for the game to load


"C:\Users\amilineni bhanudatta\Desktop\snake.exe"

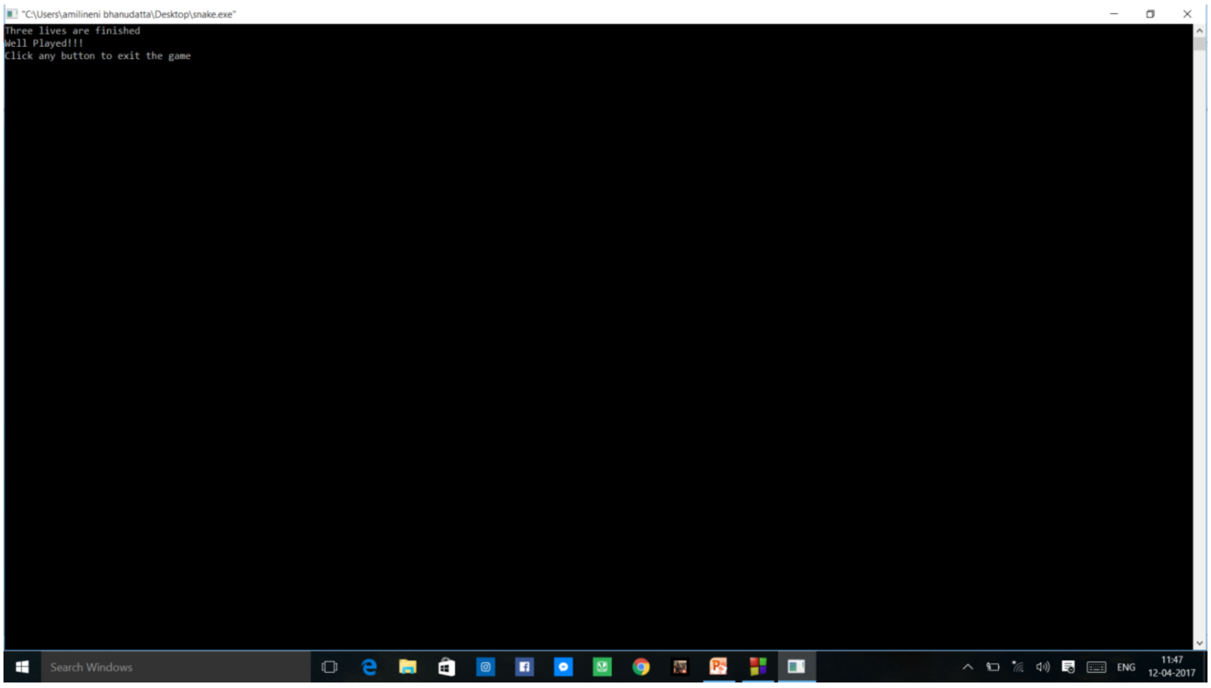
```
                           Loading.......
Please wait
```

Use the direction keys to play the game


"C:\Users\amilineni bhanudatta\Desktop\snake.exe"

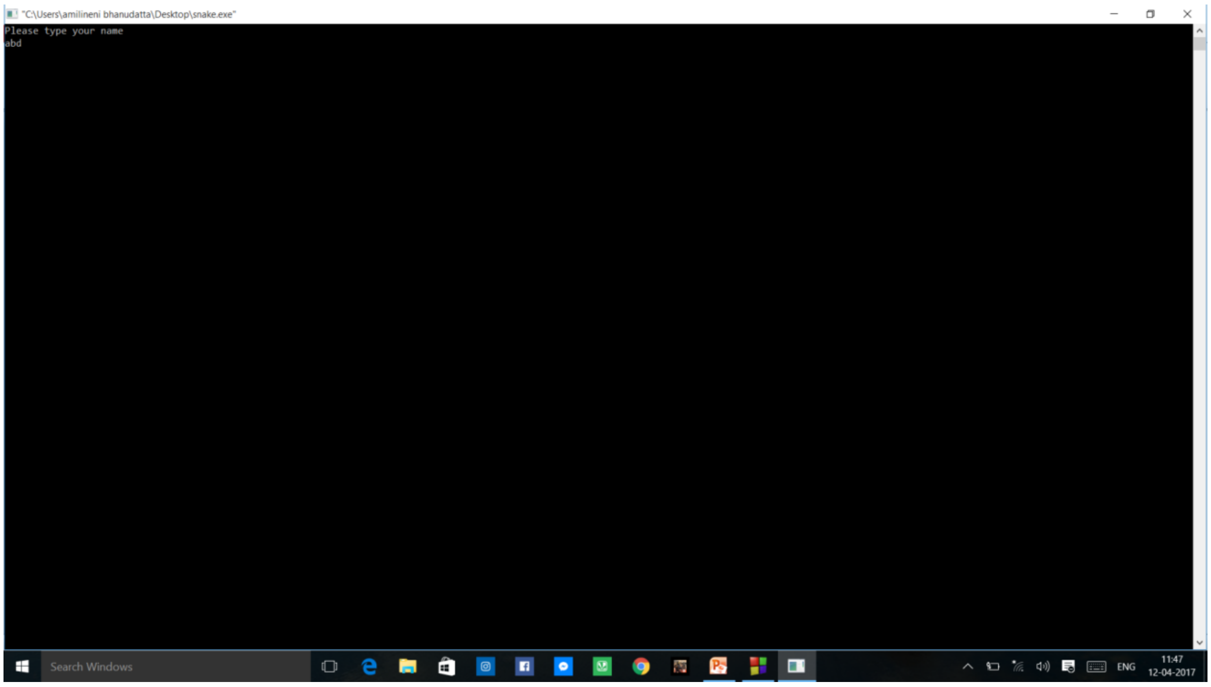
```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                                                           $
$                                                           $
$                             X        <-------             $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$                                                           $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Well played


"C:\Users\amilineni bhanudatta\Desktop\snake.exe"

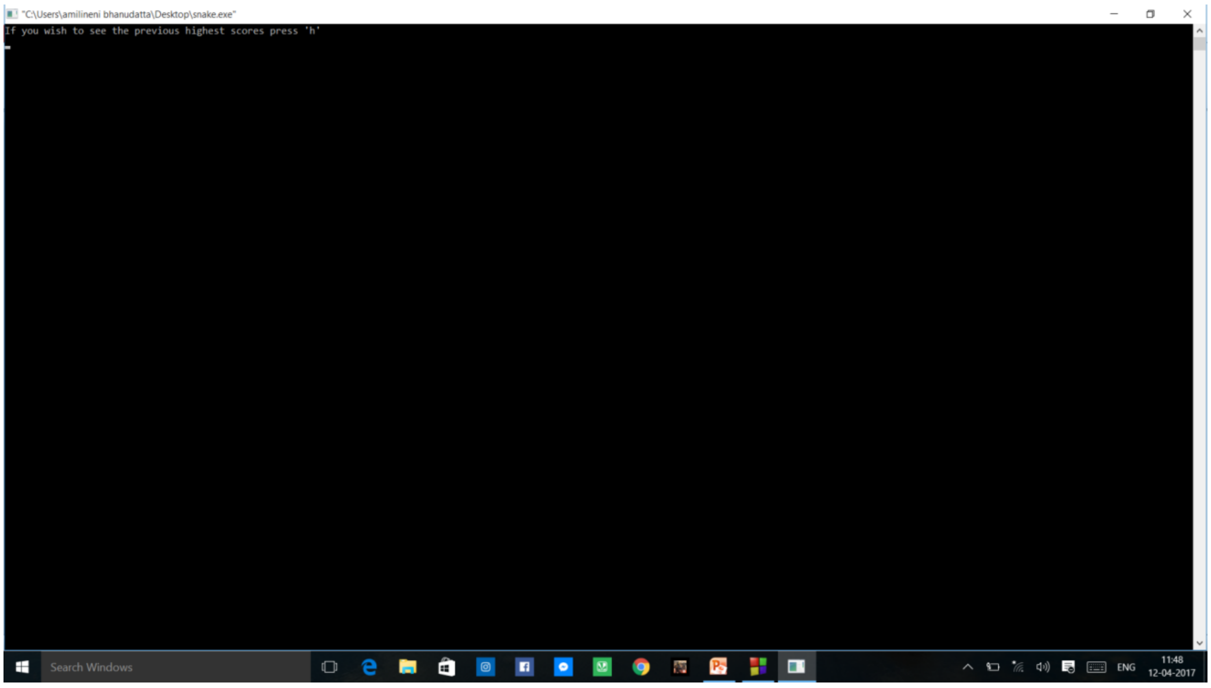
```
Three lives are finished
Well Played!!!
Click any button to exit the game
```

Submit your score



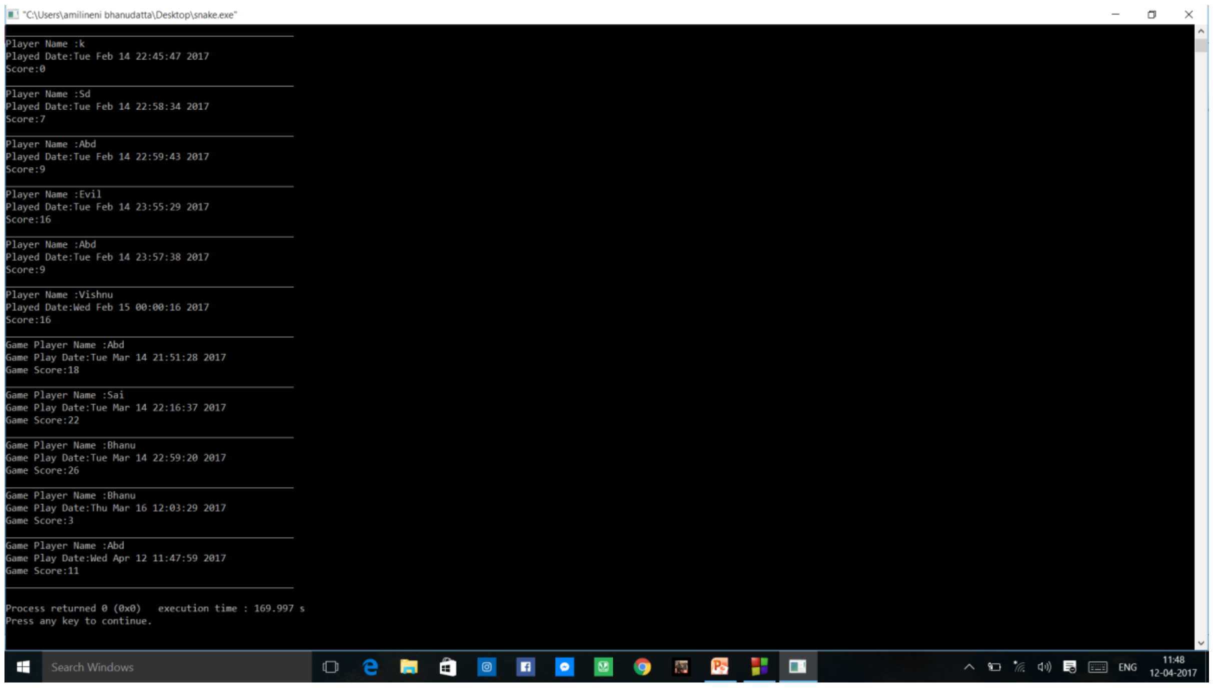
```
Please type your name
abd
```

Compare your score with others



```
If you wish to see the previous highest scores press 'h'
```

Display of previous game scores



"C:\Users\amilineni bhanudatta\Desktop\snake.exe"

```
Player Name :k
Played Date:Tue Feb 14 22:45:47 2017
Score:0

Player Name :Sd
Played Date:Tue Feb 14 22:58:34 2017
Score:7

Player Name :Abd
Played Date:Tue Feb 14 22:59:43 2017
Score:9

Player Name :Evil
Played Date:Tue Feb 14 23:55:29 2017
Score:16

Player Name :Abd
Played Date:Tue Feb 14 23:57:38 2017
Score:9

Player Name :Vishnu
Played Date:Wed Feb 15 00:00:16 2017
Score:16

Game Player Name :Abd
Game Play Date:Tue Mar 14 21:51:28 2017
Game Score:18

Game Player Name :Sai
Game Play Date:Tue Mar 14 22:16:37 2017
Game Score:22

Game Player Name :Bhanu
Game Play Date:Tue Mar 14 22:59:20 2017
Game Score:26

Game Player Name :Bhanu
Game Play Date:Thu Mar 16 12:03:29 2017
Game Score:3

Game Player Name :Abd
Game Play Date:Wed Apr 12 11:47:59 2017
Game Score:11


Process returned 0 (0x0)   execution time : 169.997 s
Press any key to continue.
```

Performance

- The execution time of the game depends on the time how much you spend on the game.
- If you make a big score and play for a longer time the execution time is more and vice versa.

References

- Tutorialspoint.com
- Books provided by the faculty for information about data structures and other stuff.
- Programised.com
- Other projects on snake game for more ideas or new features.

Snake Game Source Code

```
#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <time.h>

#include <conio.h>

#include<time.h>

#include<ctype.h>

#include <windows.h>

#include <process.h>


#define SNAKE_UP 72

#define SNAKE_DOWN 80

#define SNAKE_LEFT 75

#define SNAKE_RIGHT 77


int Length;

int Bend_snake_num;

int Len;

char KEY;

void GameRecord();

void LoadGame();

int LIFE;

void Delaygame(long double);

void Movesnake();

void Food_snake();
```

```
int Gamescore();

void print();

void goto_xy(int x, int y);

void Goto_xy(int x,int y);

void Bend_snake();

void Boarder();

void Snake_down();

void Snake_left();

void Snake_up();

void Snake_right();

void Quit_game();

int Game_score_only();


struct coordinate{

   int x;

   int y;

   int direction;

};


typedef struct coordinate coordinate;


coordinate head, bend_snake[500],food_snake,body[30];


int main()

{
```

```
char KEY;

print();

system("cls");

LoadGame();

Length=5;

head.x=25;

head.y=20;

head.direction=SNAKE_RIGHT;

Boarder();

Food_snake();

LIFE=3;

bend_snake[0]=head;

Movesnake();
```

```
    return 0;

}

void Movesnake()
{
    int a,i;

    do{

        Food_snake();
        fflush(stdin);

        Len=0;

        for(i=0;i<30;i++)

        {

            body[i].x=0;

            body[i].y=0;

            if(i==Length)

            break;
    }
    Delaygame(Length);
    Boarder();
    if(head.direction==SNAKE_RIGHT)
        Snake_right();
    else if(head.direction==SNAKE_LEFT)
        Snake_left();
    else if(head.direction==SNAKE_DOWN)
        Snake_down();
    else if(head.direction==SNAKE_UP)
        Snake_up();
    Quit_game();
}while(!kbhit());
```

```
  a=getch();

  if(a==27)

  {

    system("cls");

    exit(0);

  }
  KEY=getch();

if((KEY==SNAKE_RIGHT&&head.direction!=SNAKE_LEFT&&head.direction!=SNAKE_RIGHT)||
(KEY==SNAKE_LEFT&&head.direction!=SNAKE_RIGHT&&head.direction!=SNAKE_LEFT)||(KE
Y==SNAKE_UP&&head.direction!=SNAKE_DOWN&&head.direction!=SNAKE_UP)||(KEY==SN
AKE_DOWN&&head.direction!=SNAKE_UP&&head.direction!=SNAKE_DOWN))

  {

    Bend_snake_num++;

    bend_snake[Bend_snake_num]=head;

    head.direction=KEY;
    if(KEY==SNAKE_UP)
        head.y--;
    if(KEY==SNAKE_DOWN)
        head.y++;
    if(KEY==SNAKE_RIGHT)
        head.x++;
    if(KEY==SNAKE_LEFT)
        head.x--;
    Movesnake();
}
else if(KEY==27)
{
    system("cls");
```

```
        exit(0);

    }

    else

    {

        printf("\a");

        Movesnake();

    }
}

void goto_xy(int x, int y)
{

 COORD coord;

 coord.X = x;

 coord.Y = y;

 SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
```

```
}
void Goto_xy(int x, int y)
{
   HANDLE a;
   COORD b;
   fflush(stdout);
   b.X = x;
   b.Y = y;
   a = GetStdHandle(STD_OUTPUT_HANDLE);
   SetConsoleCursorPosition(a,b);
 }
void LoadGame(){
   int row,col,r,c,q;
   goto_xy(36,14);
   printf("LoadGameing...");
   goto_xy(30,15);
   for(r=1;r<=20;r++){
   for(q=0;q<=100000000;q++);
   printf("%c",177);}
   getch();
}
void Snake_down()
{
   int i;
   for(i=0;i<=(head.y-bend_snake[Bend_snake_num].y)&&Len<Length;i++)
```

```
    {
        Goto_xy(head.x,head.y-i);
        {
            if(Len==0)
                printf("v");
            else
                printf("-");
        }
        body[Len].x=head.x;
        body[Len].y=head.y-i;
        Len++;
    }
    Bend_snake();
    if(!kbhit())
        head.y++;
}
void Delaygame(long double k)
{
    Gamescore();
    long double i;
    for(i=0;i<=(10000000);i++);
}
void Quit_game()
{
    int i,check=0;
    for(i=4;i<Length;i++)
```

```
 {
       if(body[0].x==body[i].x&&body[0].y==body[i].y)
       {
           check++;
       }
       if(i==Length||check!=0)
           break;
   }
   if(head.x<=10||head.x>=70||head.y<=10||head.y>=30||check!=0)
   {
       LIFE--;
       if(LIFE>=0)
       {
           head.x=25;
           head.y=20;
           Bend_snake_num=0;
           head.direction=SNAKE_RIGHT;
           Movesnake();
       }
       else
       {
           system("cls");
           printf("Three lives are finished\nWell Played!!!\nClick any button to exit the
game\n");
           GameRecord();
           exit(0);
```

```
        }
    }
}
void Food_snake()
{
    if(head.x==food_snake.x&&head.y==food_snake.y)
    {
        Length++;
        time_t a;
        a=time(0);
        srand(a);
        food_snake.x=rand()%70;
        if(food_snake.x<=10)
            food_snake.x+=11;
        food_snake.y=rand()%30;
        if(food_snake.y<=10)

            food_snake.y+=11;
    }
    else if(food_snake.x==0)
    {
        food_snake.x=rand()%70;
        if(food_snake.x<=10)
            food_snake.x+=11;
        food_snake.y=rand()%30;
        if(food_snake.y<=10)
```

```
      food_snake.y+=11;
  }
}
void Snake_left()
{
  int i;
  for(i=0;i<=(bend_snake[Bend_snake_num].x-head.x)&&Len<Length;i++)
  {
    Goto_xy((head.x+i),head.y);
     {
        if(Len==0)
          printf("<");
        else
          printf("-");
    }
    body[Len].x=head.x+i;
    body[Len].y=head.y;
    Len++;
  }
  Bend_snake();
  if(!kbhit())
    head.x--;

}
void Snake_right()
{
    int i;
    for(i=0;i<=(head.x-bend_snake[Bend_snake_num].x)&&Len<Length;i++)
    {
        body[Len].x=head.x-i;
        body[Len].y=head.y;
        Goto_xy(body[Len].x,body[Len].y);
        {
            if(Len==0)
                printf(">");
            else
                printf("-");
        }
   Len++;
    }
    Bend_snake();
    if(!kbhit())
        head.x++;
}
void Bend_snake()
{
    int i,j,diff;
    for(i=Bend_snake_num;i>=0&&Len<Length;i--)
    {
            if(bend_snake[i].x==bend_snake[i-1].x)
            {
                diff=bend_snake[i].y-bend_snake[i-1].y;
        if(diff<0)
            for(j=1;j<=(-diff);j++)
            {
                body[Len].x=bend_snake[i].x;
                body[Len].y=bend_snake[i].y+j;
                Goto_xy(body[Len].x,body[Len].y);
                printf("-");
                Len++;
                if(Len==Length)
                    break;
            }
        else if(diff>0)
            for(j=1;j<=diff;j++)
            {
                body[Len].x=bend_snake[i].x;
                body[Len].y=bend_snake[i].y-j;
                Goto_xy(body[Len].x,body[Len].y);
                printf("-");
                Len++;
                if(Len==Length)
                    break;
            }
    }
else if(bend_snake[i].y==bend_snake[i-1].y)
{
    diff=bend_snake[i].x-bend_snake[i-1].x;
```

```
          if(diff<0)
              for(j=1;j<=(-diff)&&Len<Length;j++)
              {
                  body[Len].x=bend_snake[i].x+j;
                  body[Len].y=bend_snake[i].y;
                  Goto_xy(body[Len].x,body[Len].y);
                      printf("-");
                 Len++;
                 if(Len==Length)
                         break;
             }
         else if(diff>0)
             for(j=1;j<=diff&&Len<Length;j++)
             {
                 body[Len].x=bend_snake[i].x-j;
                 body[Len].y=bend_snake[i].y;
                 Goto_xy(body[Len].x,body[Len].y);
                     printf("-");
                 Len++;
                 if(Len==Length)
                     break;
             }
     }
   }
}
void Boarder()
```

```
{
   system("cls");
   int i;
   Goto_xy(food_snake.x,food_snake.y);
       printf("F");
   for(i=10;i<71;i++)
   {
       Goto_xy(i,10);
           printf("$");
       Goto_xy(i,30);
           printf("$");
   }
   for(i=10;i<31;i++)
   {
       Goto_xy(10,i);
           printf("$");
       Goto_xy(70,i);
       printf("$");
   }
}
void PRINT()
{
   //Goto_xy(10,12);
  printf("\t Mini Snake game welcomes you.(press any button to advance)\n");
  getch();
   system("cls");
   printf("\tYou should follow these Game rules:\n");
   printf("\n-> Make Use of the arrow keys given in the keyboard to make the snake move.\n\n-> Food will appear at the several coordinates of the window which you have to make the snake eat.Each time you eat food the snake will  increase by 1 element and the score will also increase.\n\n-> In this game you are given three lives. Your lives will be decreased by 1 as you come across the wall or snake's body.\n\n-> You can stop the game in its middle by tapping any key. To continue the stopped game press any other key.\n\n-> If you wish to exit the game press esc. \n");
   printf("\n\nTap any button to play the snake game...");


  if(getch()==27)
   exit(0);
}
void GameRecord(){
   char plname[20],nplname[20],cha,c;
   int i,j,px;
   FILE -info;
   info=fopen("GameRecord.txt","a+");
   getch();
   system("cls");
   printf("Enter your name\n");
   scanf("%[^\n]",plname);
   for(j=0;plname[j]!='\0';j++){
   nplname[0]=tosnake_upper(plname[0]);
   if(plname[j-1]==' '){
   nplname[j]=tosnake_upper(plname[j]);
   nplname[j-1]=plname[j-1];}
   else nplname[j]=plname[j];
```

```
   }
   nplname[j]='\0';
   fprintf(info,"Player Name :%s\n",nplname);


   time_t mytime;
  mytime = time(NULL);
  fprintf(info,"Played Date:%s",ctime(&mytime));
    fprintf(info,"Gamescore:%d\n",px=Game_score_only());
    //fprintf(info,"\nLevel:%d\n",10);
   for(i=0;i<=50;i++)
   fprintf(info,"%c",'_');
   fprintf(info,"\n");
   fclose(info);
   printf("If you wish to see the previous highest scores press 'h'\n");
   cha=getch();
   system("cls");
   if(cha=='y'){
   info=fopen("GameRecord.txt","r");
   do{
       putchar(c=getc(info));
       }while(c!=EOF);}
     fclose(info);
}
int Gamescore()
{
   int Gamescore;
  Goto_xy(20,8);

  Gamescore=Length-5;

  printf("GAMESCORE : %d",(Length-5));

  Gamescore=Length-5;

  Goto_xy(50,8);

  printf("Your Lives : %d",LIFE);

  return Gamescore;

}

int Game_score_only()

{

int Gamescore=Gamescore();

system("cls");

return Gamescore;

}

void Snake_up()

{

  int i;

  for(i=0;i<=(bend_snake[Bend_snake_num].y-head.y)&&Len<Length;i++)

  {

     Goto_xy(head.x,head.y+i);

     {

        if(Len==0)

           printf("^");

        else

           printf("-");

     }
```

```
        body[Len].x=head.x;
        body[Len].y=head.y+i;
        Len++;
    }
    Bend_snake();
    if(!kbhit())
        head.y--;
}
```