

National College of Engineering

(Affiliated to Tribhuvan University)

Talchhikhel, Lalitpur

|  |  |
| --- | --- |
| **Submitted By:** | **Submitted To:** |
| Name: Saksham Maharjan | Department of Electronics |
| Roll no: 077BCT029 | & Computer Engineering |
| Group: A2 |  |

Contents

[INTRODUCTION 1](#_Toc141825181)

[OBJECTIVE 1](#_Toc141825182)

[FUNCTIONS USED: 1](#_Toc141825183)

[1. GRAPHICS.H 1](#_Toc141825184)

[2. TIME.H 1](#_Toc141825185)

[3. WINDOWS.H 1](#_Toc141825186)

[4. initgraph() 2](#_Toc141825187)

[5. initwindow() 2](#_Toc141825188)

[6. srand() 2](#_Toc141825189)

[7. setfillstyle() 2](#_Toc141825190)

[8. bar() 3](#_Toc141825191)

[9. rand() 3](#_Toc141825192)

[10. getpixel() 3](#_Toc141825193)

[11. GetAsyncKeyState() 3](#_Toc141825194)

[12. delay() 3](#_Toc141825195)

[13. cleardevice() 3](#_Toc141825196)

[14. settextstyle() 4](#_Toc141825197)

[15. outtextxy() 4](#_Toc141825198)

[EXPECTED OUTPUT 4](#_Toc141825199)

[PROCESS 6](#_Toc141825200)

[SOURCE CODE 7](#_Toc141825201)

[OUTPUT 11](#_Toc141825202)

[CONCLUSION 13](#_Toc141825203)

# INTRODUCTION

A header file ‘graphics.h’ which is a C programming language library is commonly used for creating simple computer graphics applications. It provides a set of functions for drawing basic shapes, colors, and images on the screen. The library also allows to set up a graphics environment, to capture mouse and keyboard events for user to interact with the graphics window and simple animations.

# OBJECTIVE

The main objectives of this project are as follows:

* To learn and implement different functions of graphics.
* To interface the application of graphics to the real world.
* To familiarize with graphics and its logical coding.

# FUNCTIONS USED:

The functions used in the program are as follows:

1. GRAPHICS.H: GRAPHICS.H is a header file in C that is used to include graphics functions in a C program. It is used for drawing various shapes and as well as animate objects. It is also used to color the objects drawn.

SYNTAX: *#include<graphics.h>*

1. TIME.H: TIME.H is a C standard library header that provides functions and types for working with time and date-related operations. It allows C programs to access and manipulate time values, handle date and time information, and measure time intervals.

SYNTAX: *#include<time.h>*

1. WINDOWS.H: WINDOWS.H is a header file in the Microsoft Windows which is primarily used in C and C++ programming languages to access various Windows-specific functionalities and create Windows applications.

SYNTAX: *#include<windows.h>*

1. initgraph()function: It initializes the graphics system by loading a graphics driver from disk then puts the system into graphics modem. Initgraph() also resets all graphics settings (color, palette, current position, viewport, etc.) to their defaults, and then resets graph result to 0.

SYNTAX: *void initgraph (int\*graphdriver, int \*graphmode, char \*pathtodriver);*

1. initwindow()function: The function initializes the graphics system by opening a graphics window of the specified size. It has three parameters; width, height and title. The title parameter will be printed at the top of the window.

SYNTAX: *initwindow(width, height, title);*

1. srand()function: srand() is used to initialize random number generators. The srand() function sets the starting point for producing a series of pseudo-random integers.

SYNTAX: *srand(time(NULL));*

1. setfillstyle() function: It sets the current fill pattern and fill color. It consists of two parameters.

SYNTAX: *void setfillstyle(int pattern, int color);*

|  |  |
| --- | --- |
| **COLOR** | **INT VALUES** |
| BLACK | 0 |
| BLUE | 1 |
| GREEN | 2 |
| CYAN | 3 |
| RED | 4 |
| MAGENTA | 5 |
| BROWN | 6 |
| LIGHTGRAY | 7 |
| DARKGRAY | 8 |
| LIGHTBLUE | 9 |
| LIGHTGREEN | 10 |
| LIGHTCYAN | 11 |
| LIGHTRED | 12 |
| LIGHTMAGENTA | 13 |
| YELLOW | 14 |
| WHITE | 15 |

|  |  |
| --- | --- |
| **Pattern Fill** | **INT VALUES** |
| EMPTY\_FILL | 0 |
| SOLID\_FILL | 1 |
| LINE\_FILL | 2 |
| LTSTLASH\_FILL | 3 |
| SLASH\_FILL | 4 |
| BKSLASH\_FILL | 5 |
| LTBKSLASH\_FILL | 6 |
| HATCH\_FILL | 7 |
| XHATCH\_FILL | 8 |
| INTERLEAVE\_FILL | 9 |
| WIDE\_DOT\_FILL | 10 |
| CLOSE\_DOT\_FILL | 11 |
| USER\_FILL | 12 |

1. bar()function: It is used to draw a 2-dimensional, rectangular filled in bar.

SYNTAX: *void bar(int left, int top, int right, int bottom);*

1. rand()function: The rand() function is used to generate pseudo-random numbers. Pseudo-random numbers are numbers that appear to be random but are actually generated by a deterministic algorithm.

SYNTAX: *int rand(void);*

1. getpixel() function: getpixel() returns the color of pixel present at location (x, y).

SYNTAX: *int getpixel(int x, int y);*

1. GetAsyncKeyState()function: This function gives information about the key. i.e. It checks whether a key is pressed or not.

SYNTAX: *short GetAsynKeyState(int key);*

|  |  |
| --- | --- |
| **CODE** | **MEANING** |
| VK\_LSHIFT | LEFT-SHIFT KEY. |
| VK\_RSHIFT | RIGHT-SHIFT KEY. |
| VK\_LCONTROL | LEFT-CONTROL KEY. |
| VK\_RCONTROL | RIGHT-CONTROL KEY. |
| VK\_LMENU | LEFT-MENU KEY. |
| VK\_RMENU | RIGHT-MENU KEY. |

1. delay()function: The delay() function is used to stop the execution of the program for some period of time. It accepts a time in milliseconds to stop the execution of the program to that period of time.

SYNTAX: *delay(unsigned int);*

1. cleardevice() function: It clears the screen in graphics mode and sets the current position to (0,0). Clearing the screen consists of filling the screen with current background color.

SYNTAX: *void cleardevice();*

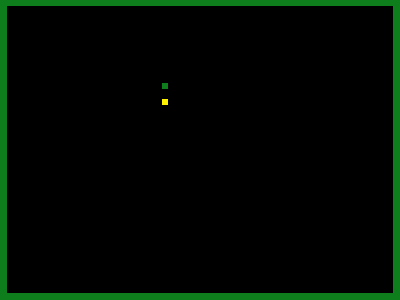
1. settextstyle() function: It is used to change the way in which the text appears. We can modify the size of text, change direction of text, and change the font of the text.

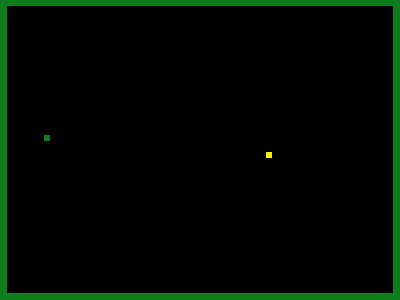
SYNTAX: *settextstyle(int font, int direction, int charsize);*

1. outtextxy() function: It is used to display the text at specified position (x,y) on the screen.

SYNTAX: *outtextxy(int x, int y, char \*string);*

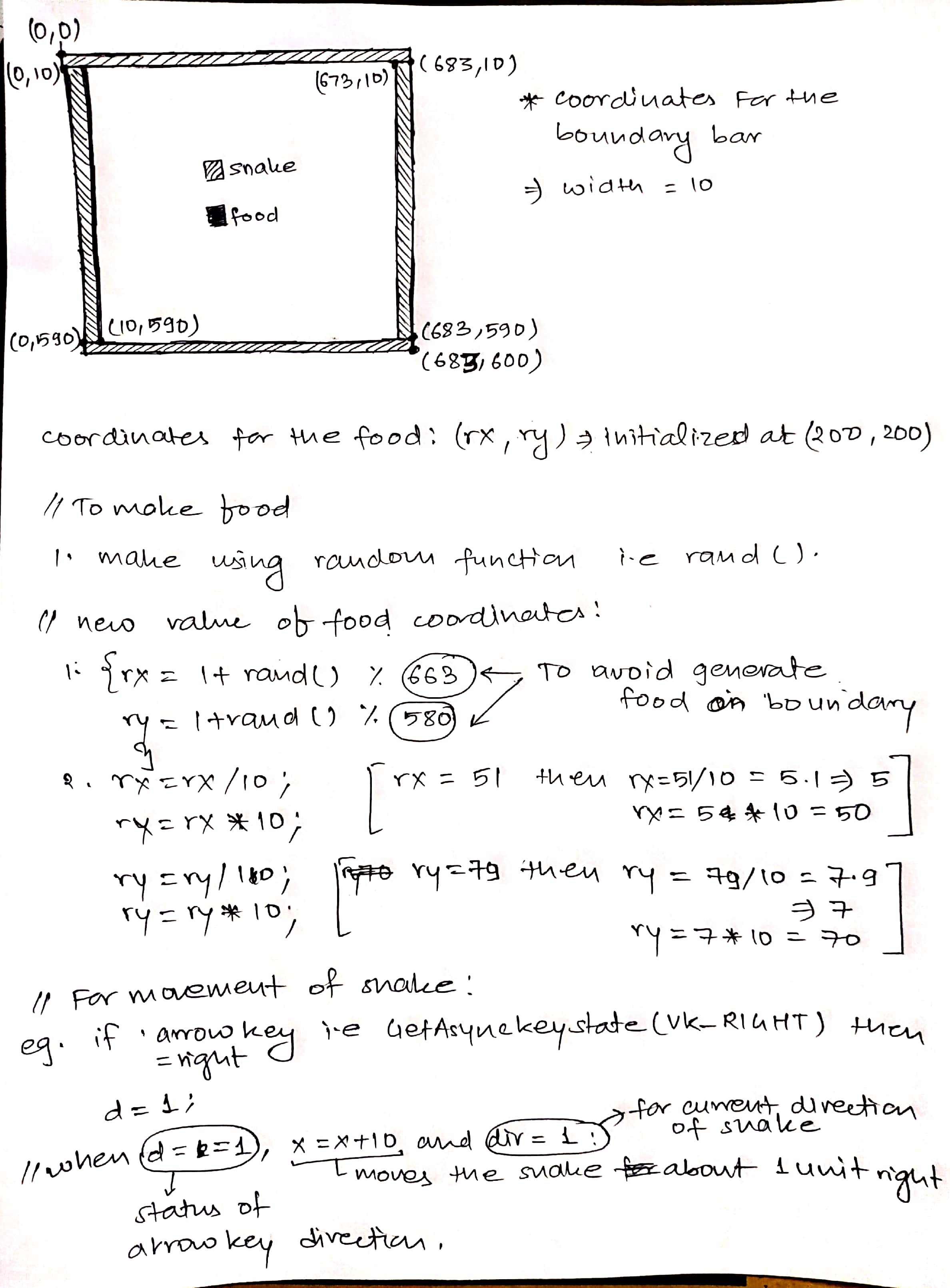
# EXPECTED OUTPUT







# PROCESS



# SOURCE CODE

#include<stdio.h>

#include<time.h>

#include<windows.h>

#include<graphics.h>

#include<stdlib.h>

void gameover();

int endfunk(int e);

int main()

{

int gd=DETECT, gm, x=200, y=200, d=1, dir=1, rx=200, ry= 200, c=0 , fx, fy;

initgraph(&gd,&gm,"");

initwindow(683,600,"SNAKEMAN"); //window resolution and name

delay(1000);

srand(time(NULL)); //for starting point of food to be random every time

setfillstyle(1,2);

for(;;)

{

setfillstyle(1,0);//screen clear to black

bar(0,0,683,600);

setfillstyle(1,2);//boundary color for snake

//boundary bars

bar(0,0,683,10);

bar(0,590,683,600);

bar(0,10,10,590);

bar(673,10,683,590);

//to make food

if(x == rx && y == ry)

{

c = c + 1; //food counter for score

setfillstyle(1,0); //color to erase the previous food

bar(rx,ry,rx+10,ry+10); //previous food

do

{

rx = (1 + rand() % 663);

ry = (1 + rand() % 580);

}while(getpixel(rx,ry) != 0 && rx > 10 && ry > 10);

rx = rx / 10;

rx = rx \* 10;

ry = ry / 10;

ry = ry \* 10;

setfillstyle(1,14); // color for when snake reach the food

}

setfillstyle(1,14); // color for when new food is displayed

bar(rx,ry,rx+10,ry+10); //new food

setfillstyle(1,2);

//arrow keys

if(GetAsyncKeyState(VK\_RIGHT))

{

d = 1;

}

else if(GetAsyncKeyState(VK\_LEFT))

{

d = 2;

}

else if(GetAsyncKeyState(VK\_UP))

{

d = 3;

}

else if(GetAsyncKeyState(VK\_DOWN))

{

d = 4;

}

else

{

d = 0;

}

switch(d)

{

case 0: //when no arrow key is pressed

if(dir == 1)

{

x = x+10;

}

else if(dir == 2)

{

x = x-10;

}

else if(dir == 3)

{

y = y - 10;

}

else if(dir == 4)

{

y = y + 10;

}

else

{

d = 0;

}

break;

case 1: //right key

x = x + 10;

dir = 1;

break;

case 2: //left key

x = x - 10;

dir = 2;

break;

case 3: //up key

y = y - 10;

dir = 3;

break;

case 4: //down key

y = y + 10;

dir = 4;

break;

}

bar(x,y,x+10,y+10); //next move of snake

delay(100);

if(x >= 683 || x <= 0 || y <= 0 || y >= 600)// when snake cross the boundary

{

cleardevice();

gameover();

delay(2000);

endfunk(c);

break;

}

}

}

void gameover()

{

setfillstyle(1,WHITE);

settextstyle(3,0,5);

outtextxy((getmaxx()/2)-130,(getmaxy()/2)-50,"Game Over");

}

int endfunk(int e)

{

e=e-1;

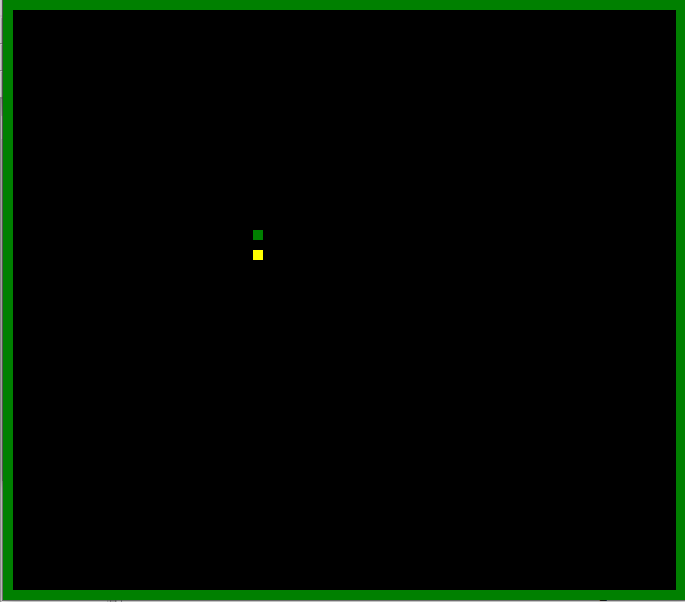
system("cls"); //to clear the console window

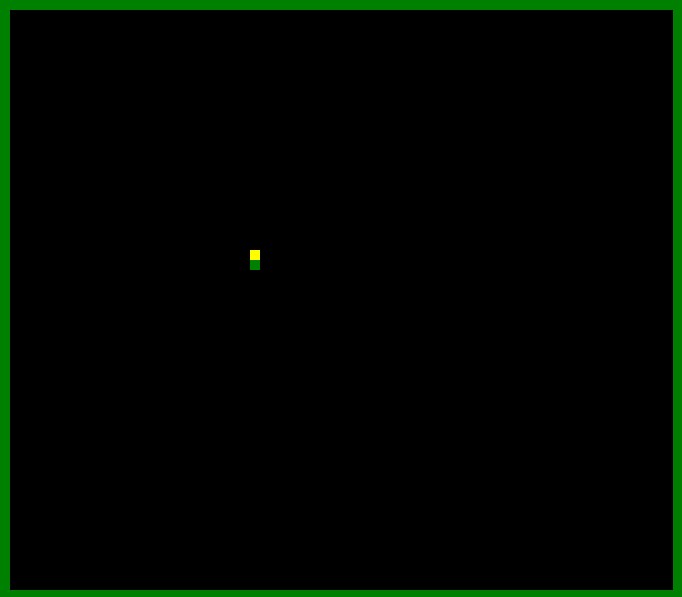
printf("You died outside the boundary!!!\n");

printf("Your score is : %d\n", e);

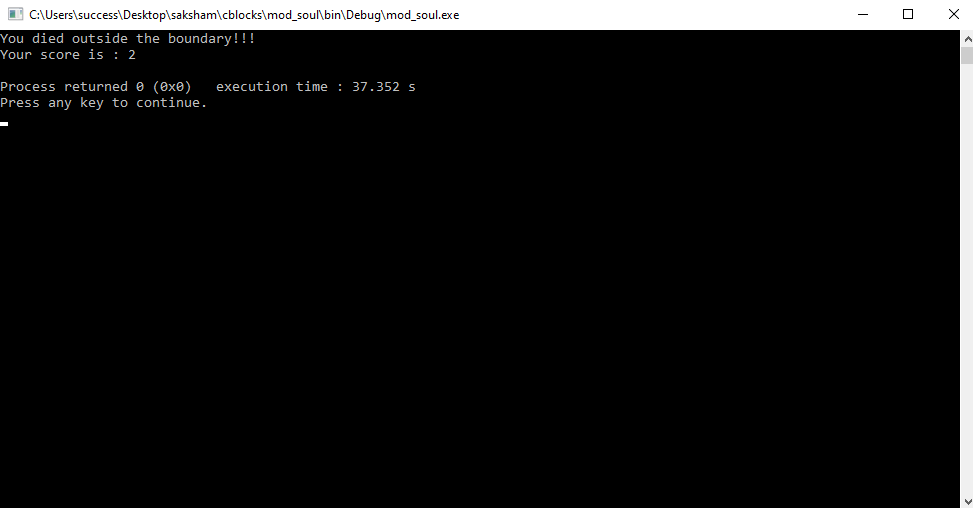
}

# OUTPUT









# CONCLUSION

Thus, using various inbuilt graphics functions of header file graphics.h, we were able to create a snake game. We learnt about these functions, implemented them in a C program and successfully operated the program to get a graphics image as output. Many problems that arises during execution of the program were fixed by thorough debugging. In this way, we got familiarized with various graphics functions and learnt about their uses and implementation.