ASSIGNMENT NO :- 1

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE

#include<iostream.h>

#include<dos.h>

#include<conio.h> #include<graphics.h>

class xyz

{

public:

void scanline(int [] ,int [],int);

};

void xyz :: scanline(int xx[10],int yy[10],int n)

{

int i,k,temp,ymax=480,ymin=0,inx[50],y,cnt ; float m[50],dx,dy; setcolor(BLUE); for(i=0;i<n;i++)

{

if(yy[i]>=ymax) ymax=yy[i];

if(yy[i]<=ymin) ymin=yy[i];

dx = xx[i+1] - xx[i]; dy = yy[i+1]-yy[i];

if(dx == 0)

m[i] = 0;

if(dy == 0)

m[i] =1;

if(dx!=0 && dy!=0) m[i]= (float) (dx/dy);

}

for(y=ymin;y<=ymax;y++)

{

int cnt=0; for(i=0;i<n;i++)

{

if((yy[i]>y&&yy[i+1]<=y) || ( yy[i]<=y&&yy[i+1]>y))

{

inx[cnt]=(xx[i]+(m[i] \* (y-yy[i])));

cnt++;

}

}

for(k=0;k<cnt-1;k++)

{

for(i=0;i<cnt-1;i++)

{

if(inx[i]>inx[i+1])

{

temp = inx[i]; inx[i] = inx[i+1]; inx[i+1] = temp;

}

}

}

for(i=0;i<cnt-1;i+=2)

{

line(inx[i],y,inx[i+1]+1,y); delay(100);

}

}

}

int main()

{

int xx[10],yy[10],n,i=0,x,y,gd=DETECT,gm=0; xyz s;

cout<<"Enter No. of vertices::"; cin>>n; for(i=0;i<n;i++)

{

cout<<"\n Enter vertices of poly::"; cin>>xx[i]; cin>>yy[i];

}

xx[i] = xx[0]; yy[i] = yy[0];

initgraph(&gd,&gm,"//TURBOC3//BGI"); for(i=0;i<=n-1;i++)

{

delay(200); line(xx[i],yy[i],xx[i+1],yy[i+1]);

}

s.scanline(xx,yy,n);

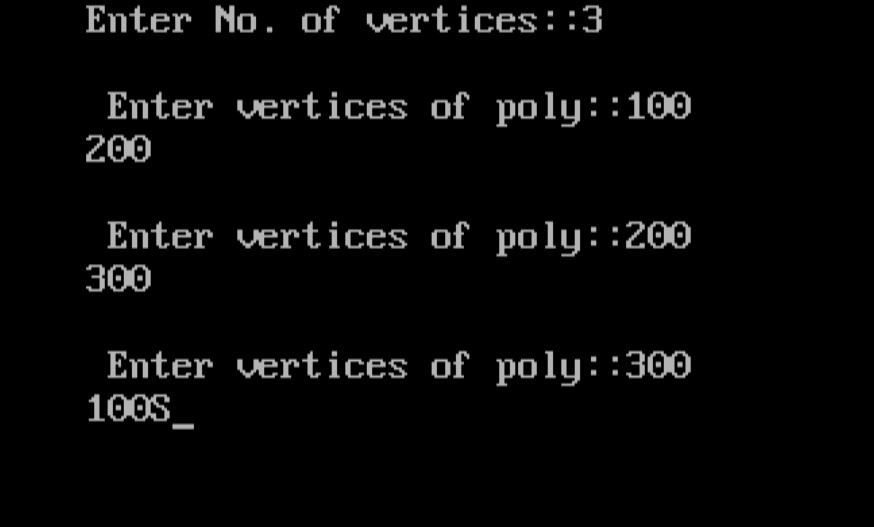
getch();

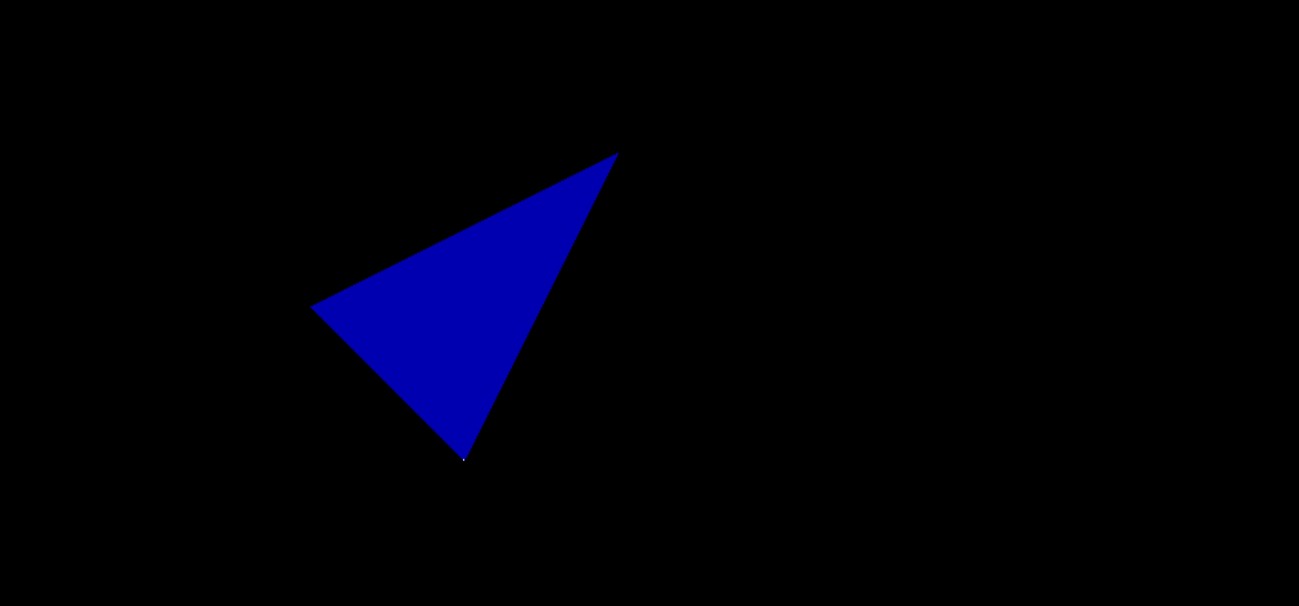
// capture("abc.bmp"); closegraph(); return 0;

}

:

**OUTPUT :**





ASSIGNMENT NO :- 2

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<conio.h>

#include<iostream.h>

#include<graphics.h>

static int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xmin,ymin,xmax,ymax;

int getcode(int x,int y){

int code = 0;

//Perform Bitwise OR to get outcode

if(y > ymax) code |=TOP;

if(y < ymin) code |=BOTTOM;

if(x < xmin) code |=LEFT;

if(x > xmax) code |=RIGHT;

return code;

}

int main()

{

int gdriver = DETECT,gmode;

initgraph(&gdriver,&gmode,"//TURBOC3//BGI ");

setcolor(WHITE);

cout<<"Enter Enter windows min and max values: ";

cin>>xmin>>ymin>>xmax>>ymax;

rectangle(xmin,ymin,xmax,ymax);

int x1,y1,x2,y2;

cout<<"Enter the endpoints of the line: ";

cin>>x1>>y1>>x2>>y2;

line(x1,y1,x2,y2);

getch();

int outcode1=getcode(x1,y1), outcode2=getcode(x2,y2);

int accept = 0; //decides if line is to be drawn

while(1){

float m =(float)(y2-y1)/(x2-x1);

//Both points inside. Accept line

if(outcode1==0 && outcode2==0)

{

accept = 1;

break;

}

//AND of both codes != 0.Line is outside. Reject line

else if((outcode1 & outcode2)!=0)

{

break;

}

else

{

int x,y;

int temp;

//Decide if point1 is inside, if not, calculate intersection

if(outcode1==0)

temp = outcode2;

else

temp = outcode1;

//Line clips top edge

if(temp & TOP){

x = x1+ (ymax-y1)/m;

y = ymax;

}

else if(temp & BOTTOM){ //Line clips bottom edge

x = x1+ (ymin-y1)/m;

y = ymin;

}else if(temp & LEFT){ //Line clips left edge

x = xmin;

y = y1+ m\*(xmin-x1);

}else if(temp & RIGHT){ //Line clips right edge

x = xmax;

y = y1+ m\*(xmax-x1);

}

if(temp == outcode1){

x1 = x;

y1 = y;

outcode1 = getcode(x1,y1);

}else{

x2 = x;

y2 = y;

outcode2 = getcode(x2,y2);

}

}

}

cout<<"After clipping:";

if(accept)

cleardevice();

rectangle(xmin,ymin,xmax,ymax);

setcolor(RED);

line(x1,y1,x2,y2);

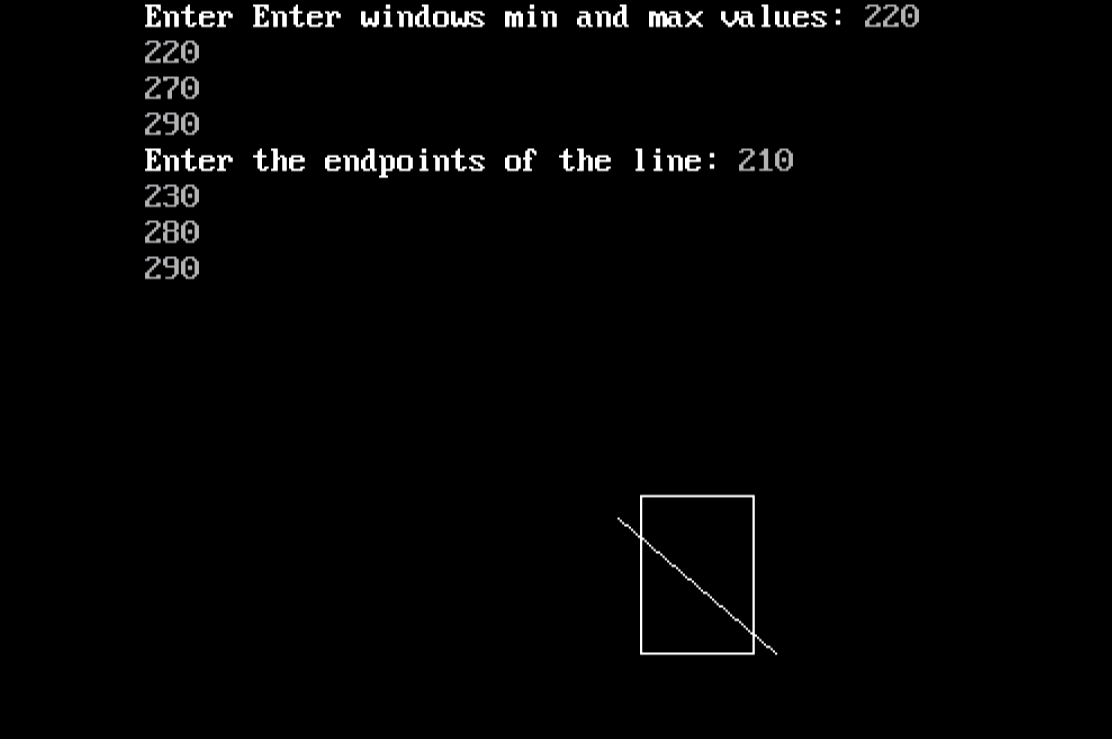
getch();

closegraph();

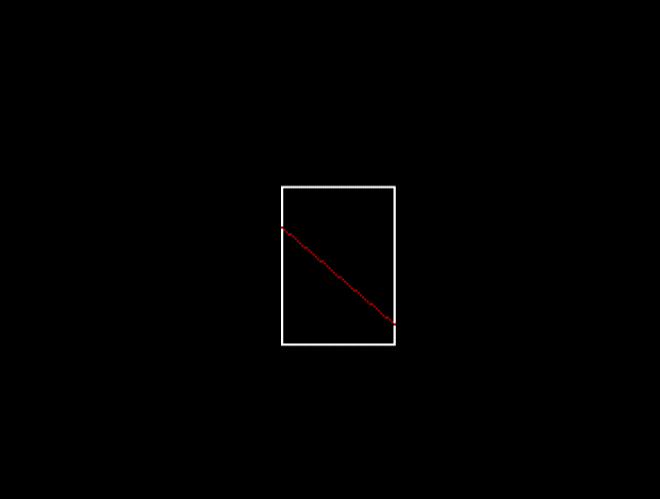
return 0;

}

**OUTPUT :**



**After Clipping**



ASSIGNMENT NO :- 3

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<iostream.h>

#include<dos.h>

#include<conio.h>

#include<graphics.h>

void ddaAlg(int x1,int y1,int x2,int y2)

{

int dx=x2-x1;

int dy=y2-y1;

int steps=dx>dy?dx:dy;

float xInc=dx/(float)steps;

float yInc=dy/(float)steps;

float x=x1;

float y=y1;

for(int i=0;i<=steps;i++)

{

putpixel(x,y,14);

x+=xInc;

y+=yInc;

}

}

void display(int xc,int yc,int x,int y)

{

putpixel(xc+x, yc+y, 3);

putpixel(xc-x, yc+y, 3);

putpixel(xc+x, yc-y, 3);

putpixel(xc-x, yc-y, 3);

putpixel(xc+y, yc+x, 3);

putpixel(xc-y, yc+x, 3);

putpixel(xc+y, yc-x, 3);

putpixel(xc-y, yc-x, 3);

}

void CircleB(int x1,int y1,int r)

{

int x=0,y=r;

int d=3-2\*r;

display(x1,y1,x,y);

while(y>=x)

{

x++;

if(d>0)

{

y--;

d=d+4\*(x-y)+10;

}

else

{

d=d+4\*x+6;

}

display(x1,y1,x,y);

}

}

int main()

{

int gd=DETECT, gm;

initgraph(&gd,&gm,"//TURBOC3//BGI");

CircleB(150,180,57);

CircleB(150,180,57/2);

ddaAlg(102,150,198,150);

ddaAlg(102,150,150,236);

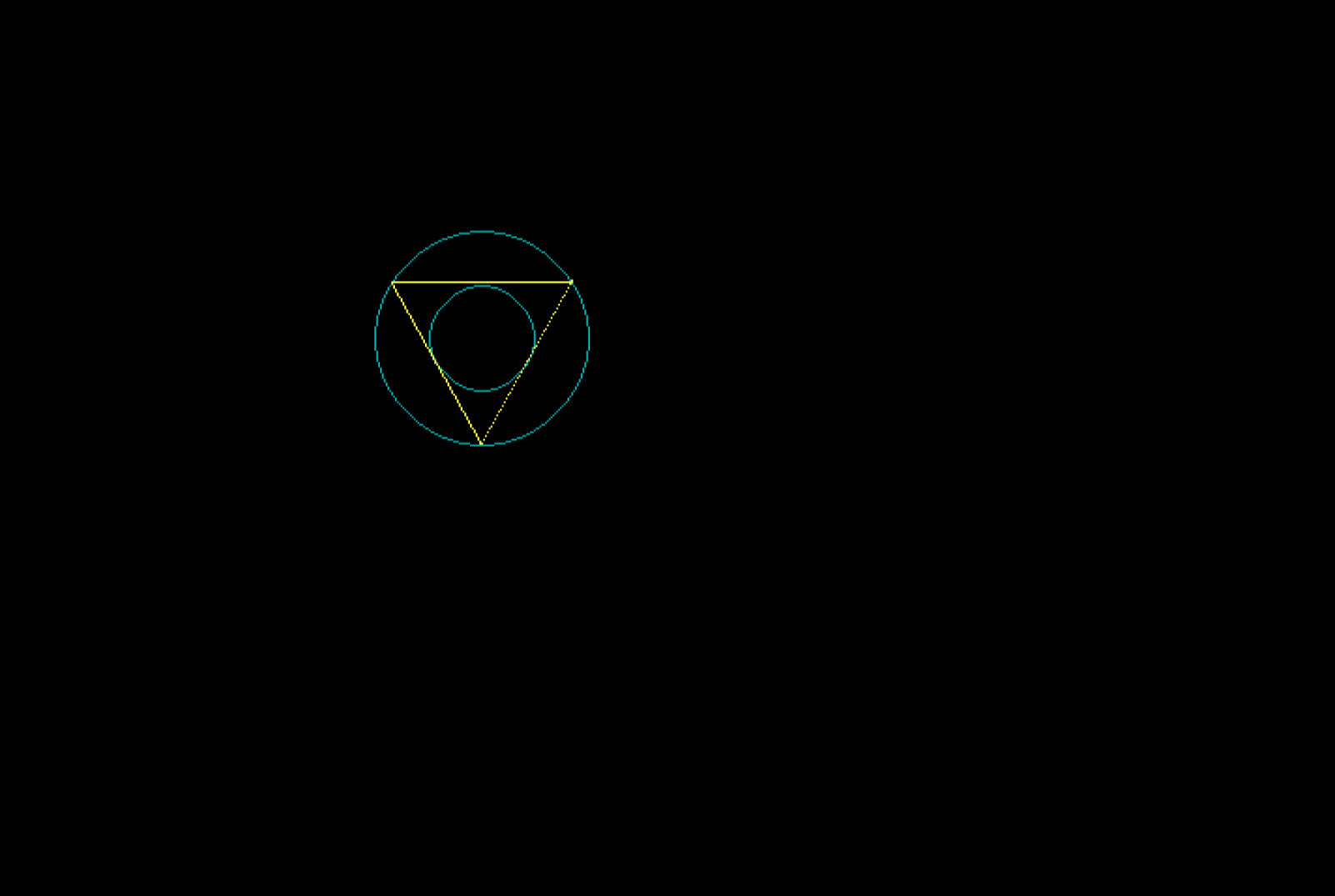
ddaAlg(150,236,198,150);

getch();

closegraph();

return 0;

}



ASSIGNMENT NO :- 4

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<iostream.h>

#include<dos.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

class transform

{

public:

int m,a[20][20],c[20][20];

int i,j,k;

public:

void object();

void accept();

void reflect();

void operator \*(float b[20][20])

{

for(int i=0;i<m;i++)

{

for(int j=0;j<m;j++)

{

c[i][j]=0;

for(int k=0;k<m;k++)

{

c[i][j]=c[i][j]+(a[i][k]\*b[k][j]);

}

}

}

}

};

void transform::object()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"//TURBOC3//BGI");

line(300,0,300,600);

line(0,300,600,300);

for( i=0;i<m-1;i++)

{

line(300+a[i][0],-a[i][1]+300,300+a[i+1][0],300-a[i+1][1]);

}

line(300+a[0][0],300-a[0][1],300+a[i][0],300-a[i][1]);

for( i=0;i<m-1;i++)

{

line(300+c[i][0],300-c[i][1],300+c[i+1][0],300-c[i+1][1]);

}

line(300+c[0][0],300-c[0][1],300+c[i][0],300-c[i][1]);

int temp;

cout << "Press 1 to Continue";

cin >> temp;

closegraph();

}

void transform::accept()

{

cout<<"\n" ;

cout<<"Enter the Number Of Edges:";

cin>>m;

cout<<"\nEnter The Coordinates :";

for(int i=0;i<m;i++)

{

for(int j=0;j<3;j++)

{

if(j>=2)

a[i][j]=1;

else

cin>>a[i][j];

}

}

}

int main()

{

int ch,tx,ty,sx,sy;

float deg,theta,b[20][20];

transform t;

t.accept();

do

{

cout<<"\nEnter your choice";

cout<<"\n1.Translation"

"\n2.Scaling"

"\n3.Rotation";

cin>>ch;

switch(ch)

{

case 1:cout<<"\nTANSLATION OPERATION\n";

cout<<"Enter value for tx and ty:";

cin>>tx>>ty;

b[0][0]=b[2][2]=b[1][1]=1;

b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;

b[2][0]=tx;

b[2][1]=ty;

t \* b;

t.object();

break;

case 2:cout<<"\nSCALING OPERATION\n";

cout<<"Enter value for sx,sy:";

cin>>sx>>sy;

b[0][0]=sx;

b[1][1]=sy;

b[0][1]=b[0][2]=b[1][0]=b[1][2]=0;

b[2][0]=b[2][1]=0;

t \* b;

t.object();

break;

case 3:cout<<"\nROTATION OPERATION\n";

cout<<"Enter value for angle:";

cin>>deg;

theta=deg\*(3.14/180);

b[0][0]=b[1][1]=cos(theta);

b[0][1]=sin(theta);

b[1][0]=sin(-theta);

b[0][2]=b[1][2]=b[2][0]=b[2][1]=0;

b[2][2]=1;

t \* b;

t.object();

break;

default:

cout<<"\nInvalid choice";

break;

}

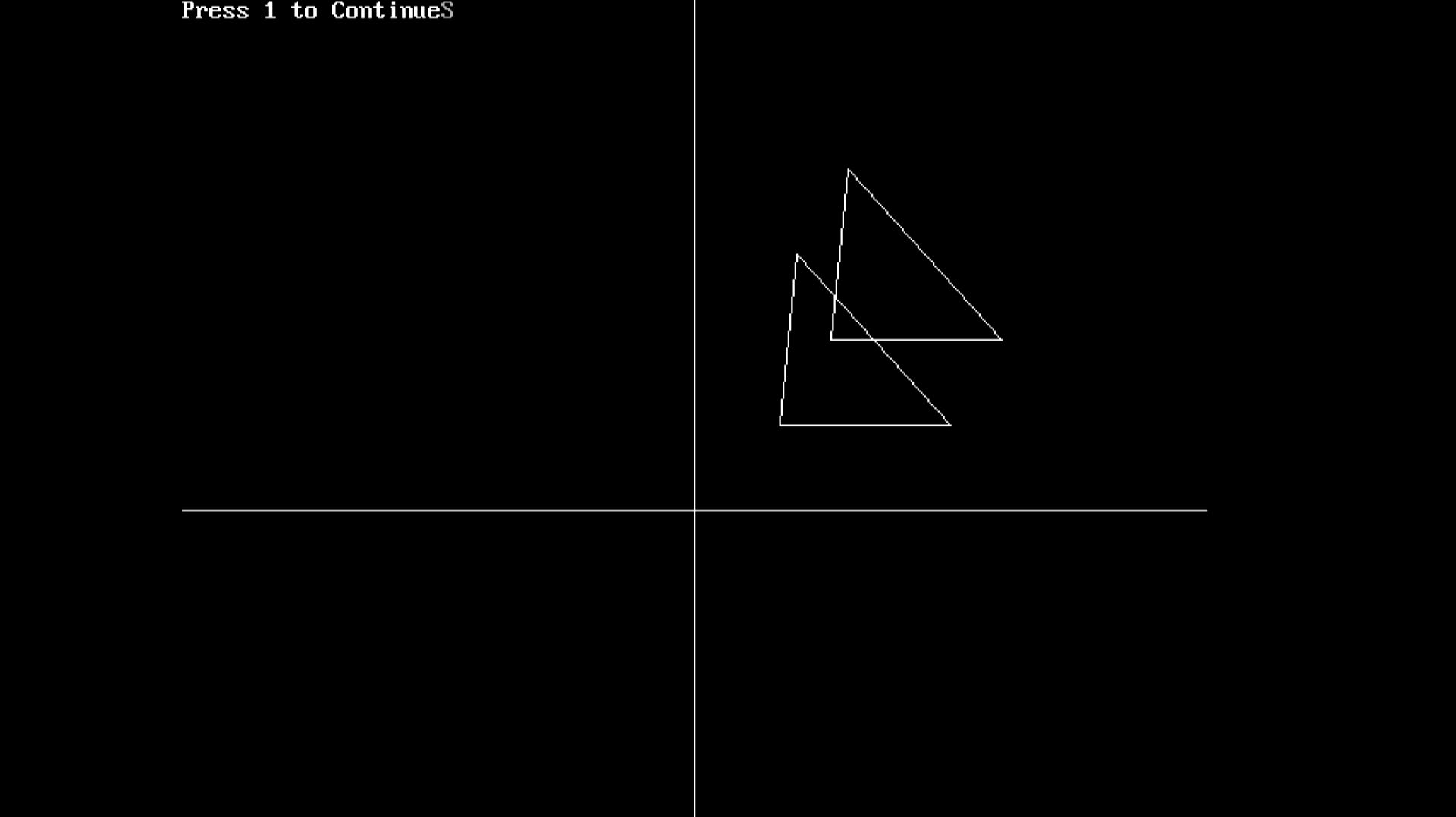
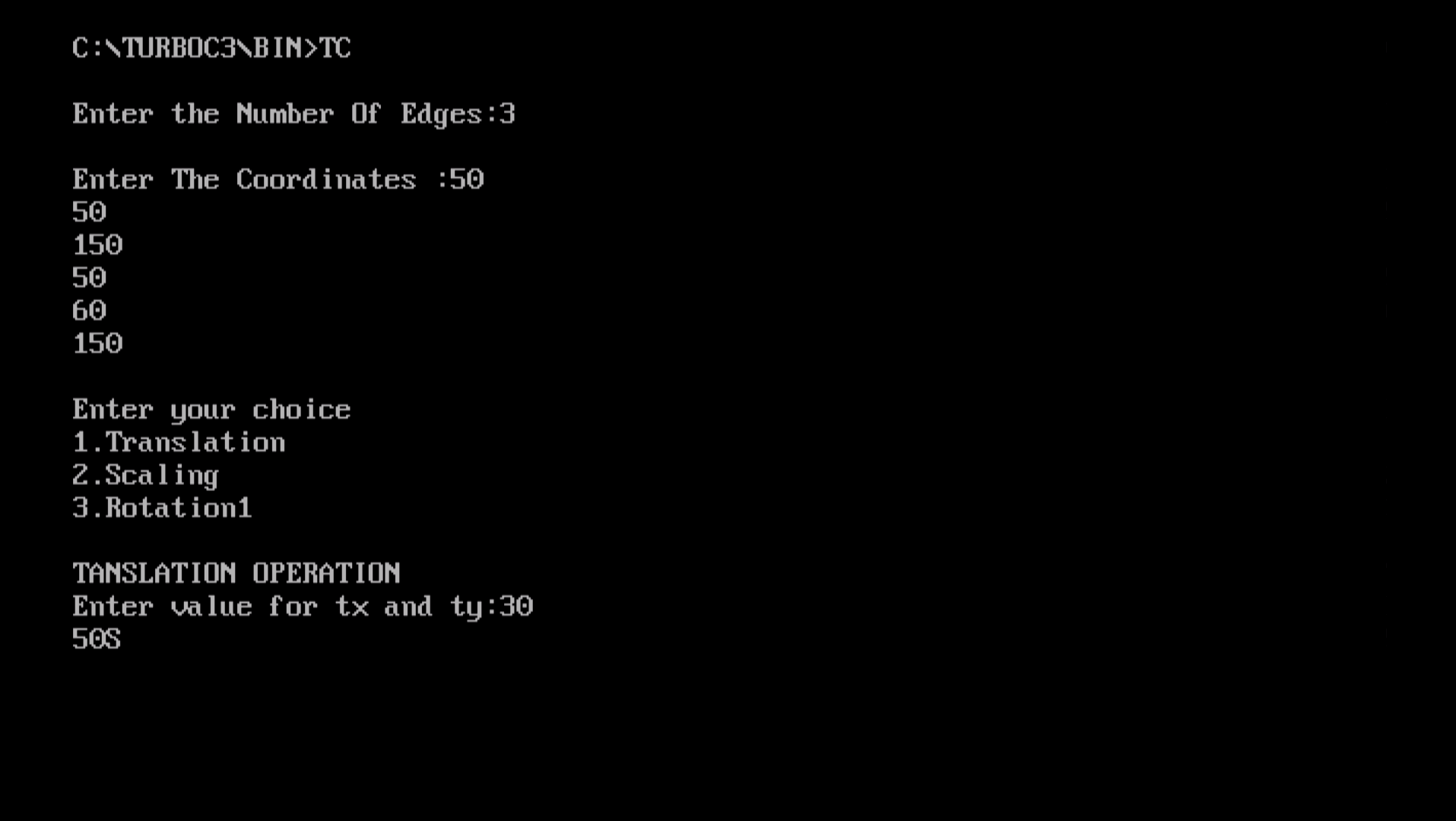
}while(ch!=4);

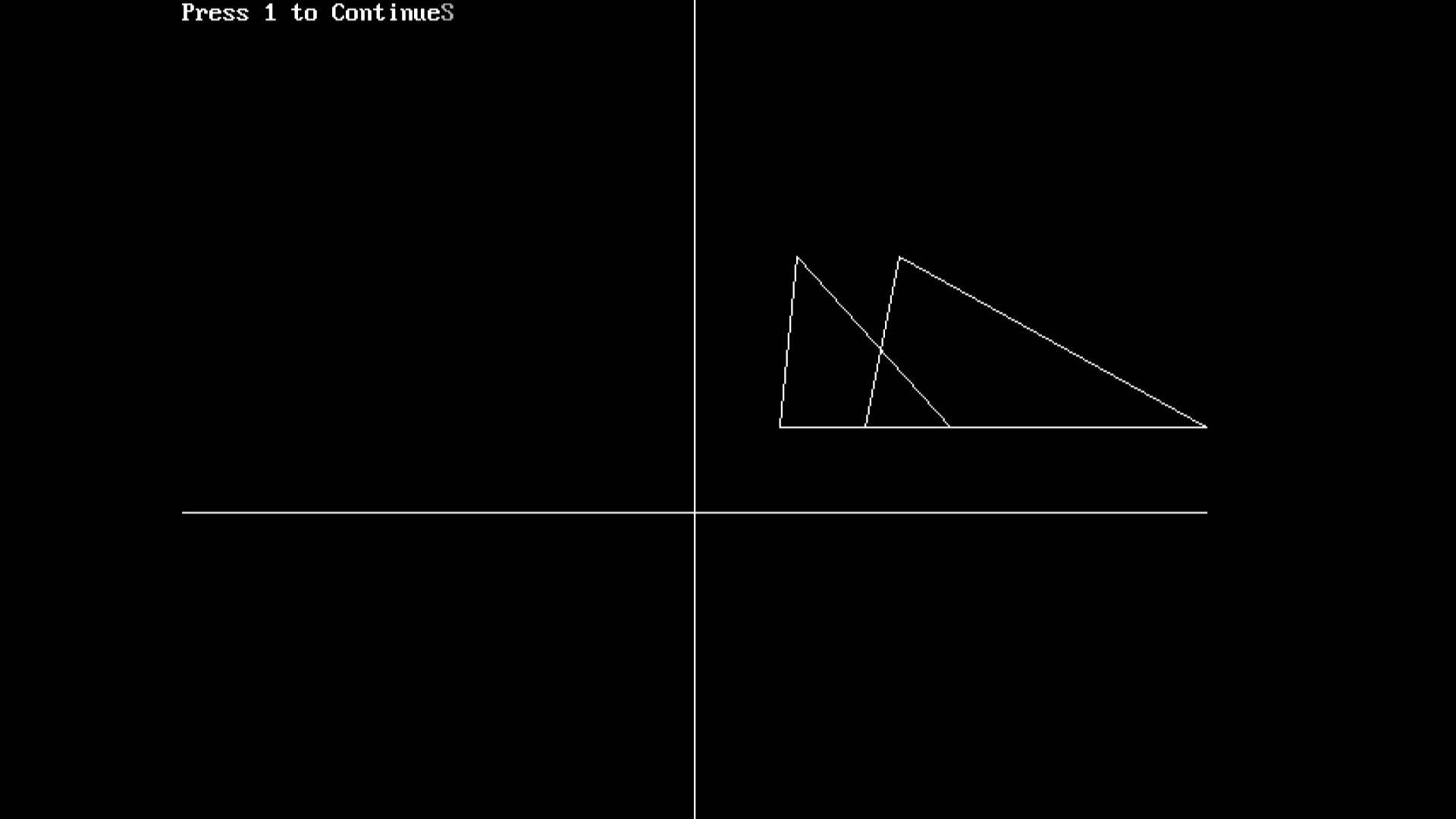
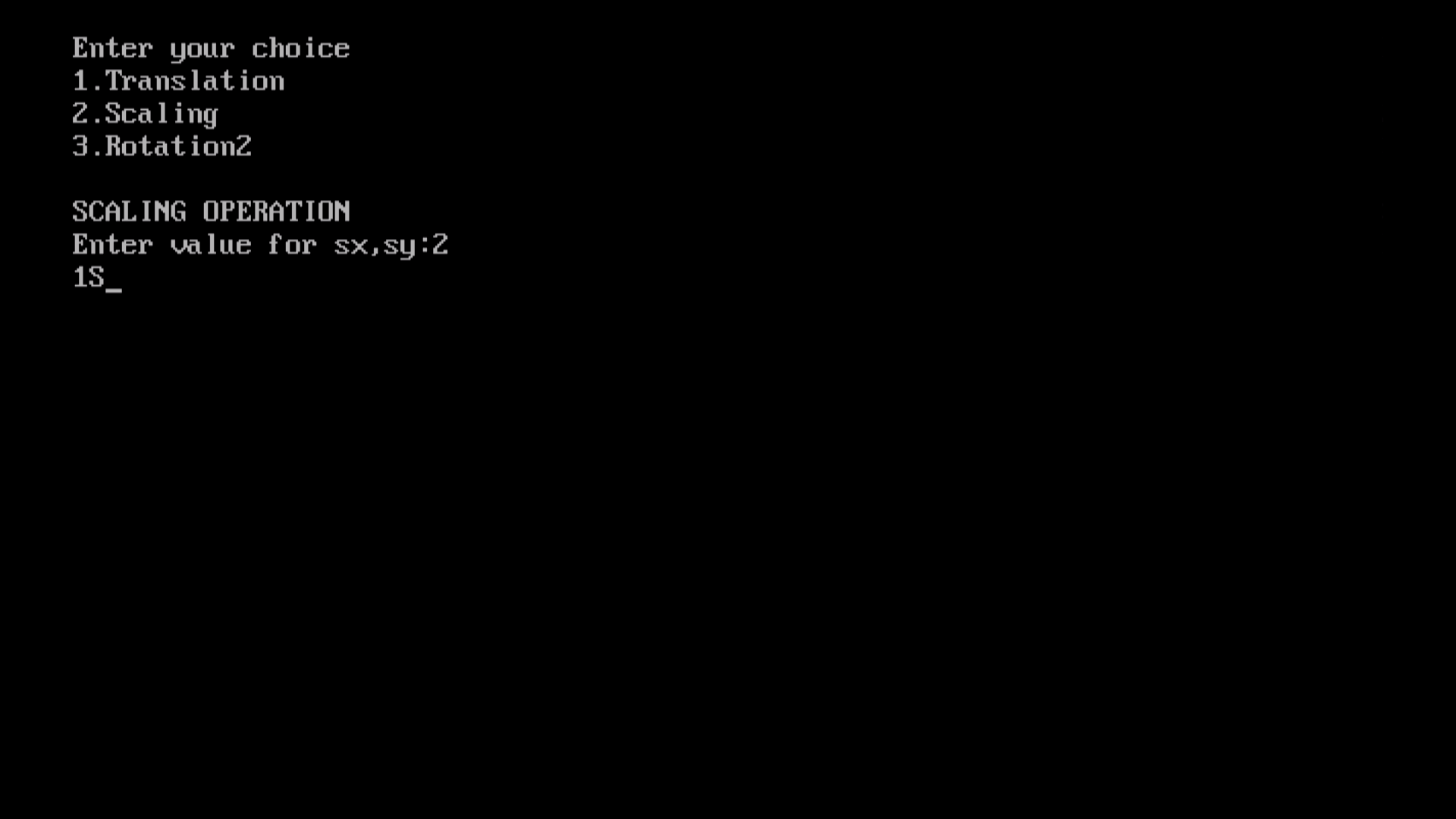
getch();

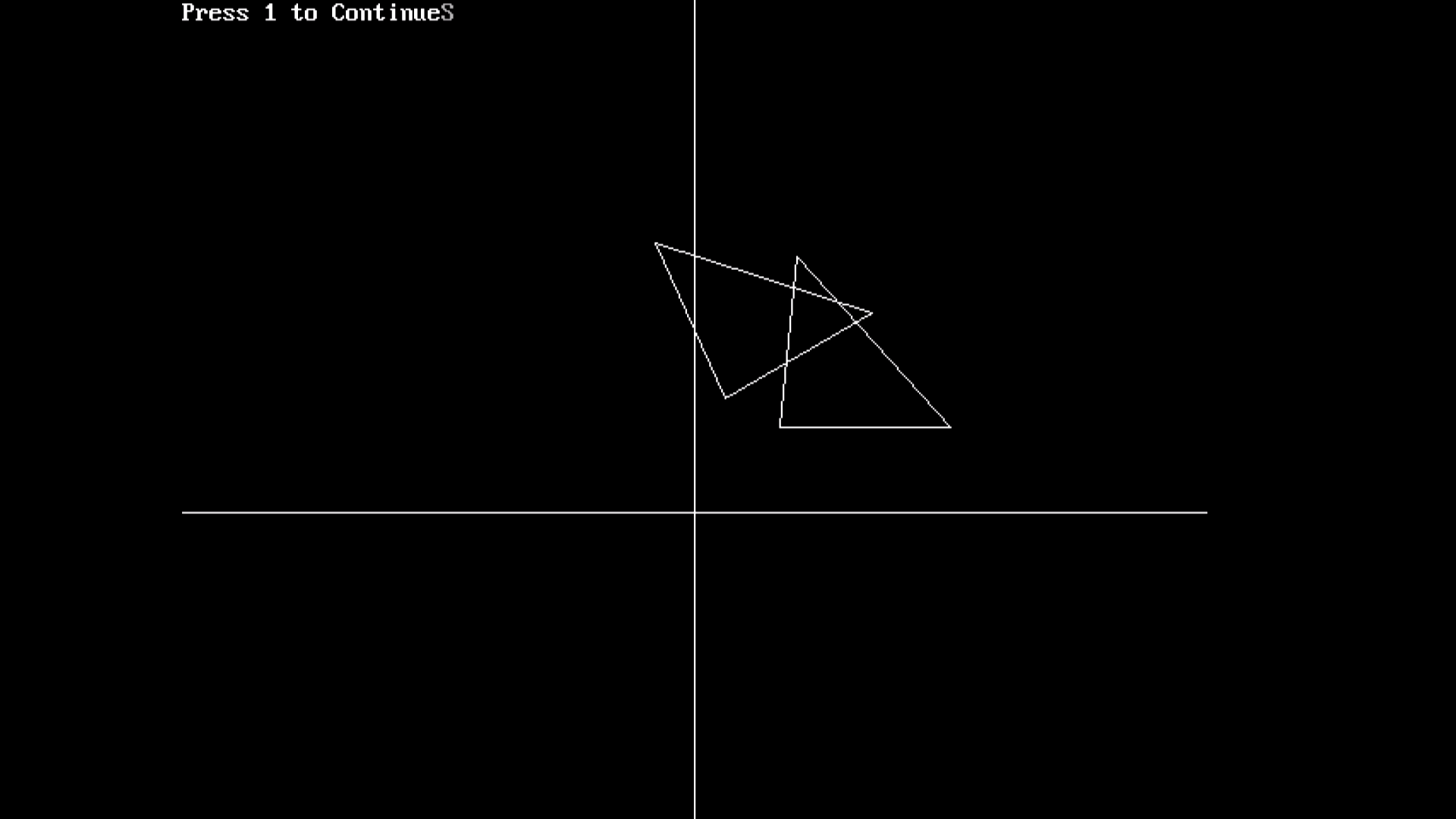
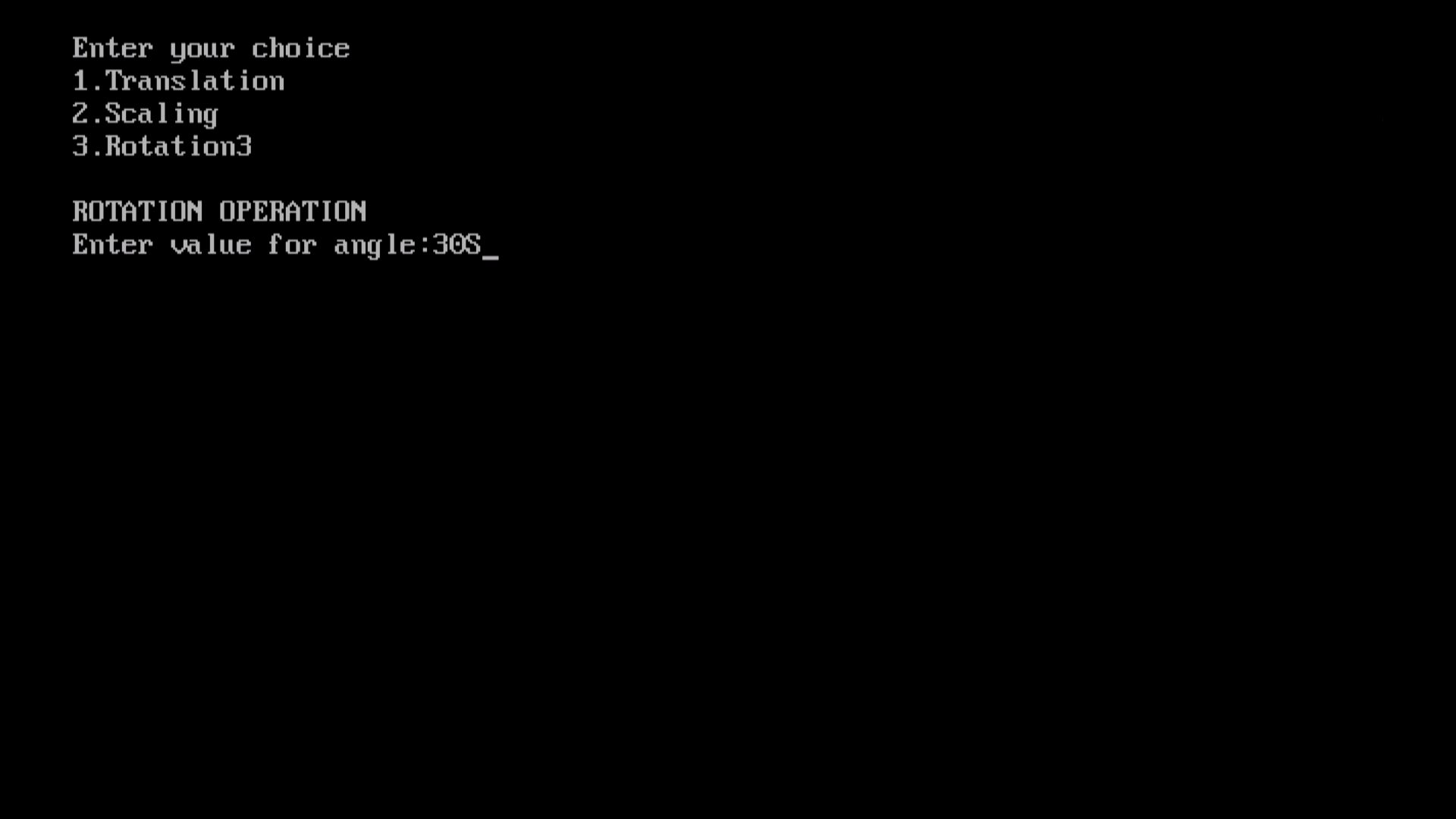
return 0;

}

**OUTPUT :**







ASSIGNMENT NO :- 5

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<iostream.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

#include<dos.h>

void move(int j,int h,int &x,int &y)

{

if(j==1)

y-=h;

else if(j==2)

x+=h;

else if(j==3)

y+=h;

else if(j==4)

x-=h;

lineto(x,y);

}

void hilbert(int r,int d,int l,int u,int i,int h,int &x,int &y)

{

if(i>0)

{

i--;

hilbert(d,r,u,l,i,h,x,y);

move(r,h,x,y);

hilbert(r,d,l,u,i,h,x,y);

move(d,h,x,y);

hilbert(r,d,l,u,i,h,x,y);

move(l,h,x,y);

hilbert(u,l,d,r,i,h,x,y);

}

}

int main()

{

int n,x1,y1;

int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;

cout<<"\nGive the value of n: ";

cin>>n;

x=x0;y=y0;

int gd=DETECT,gm;

initgraph(&gd,&gm,"//TURBOC3//BGI");

moveto(x,y);

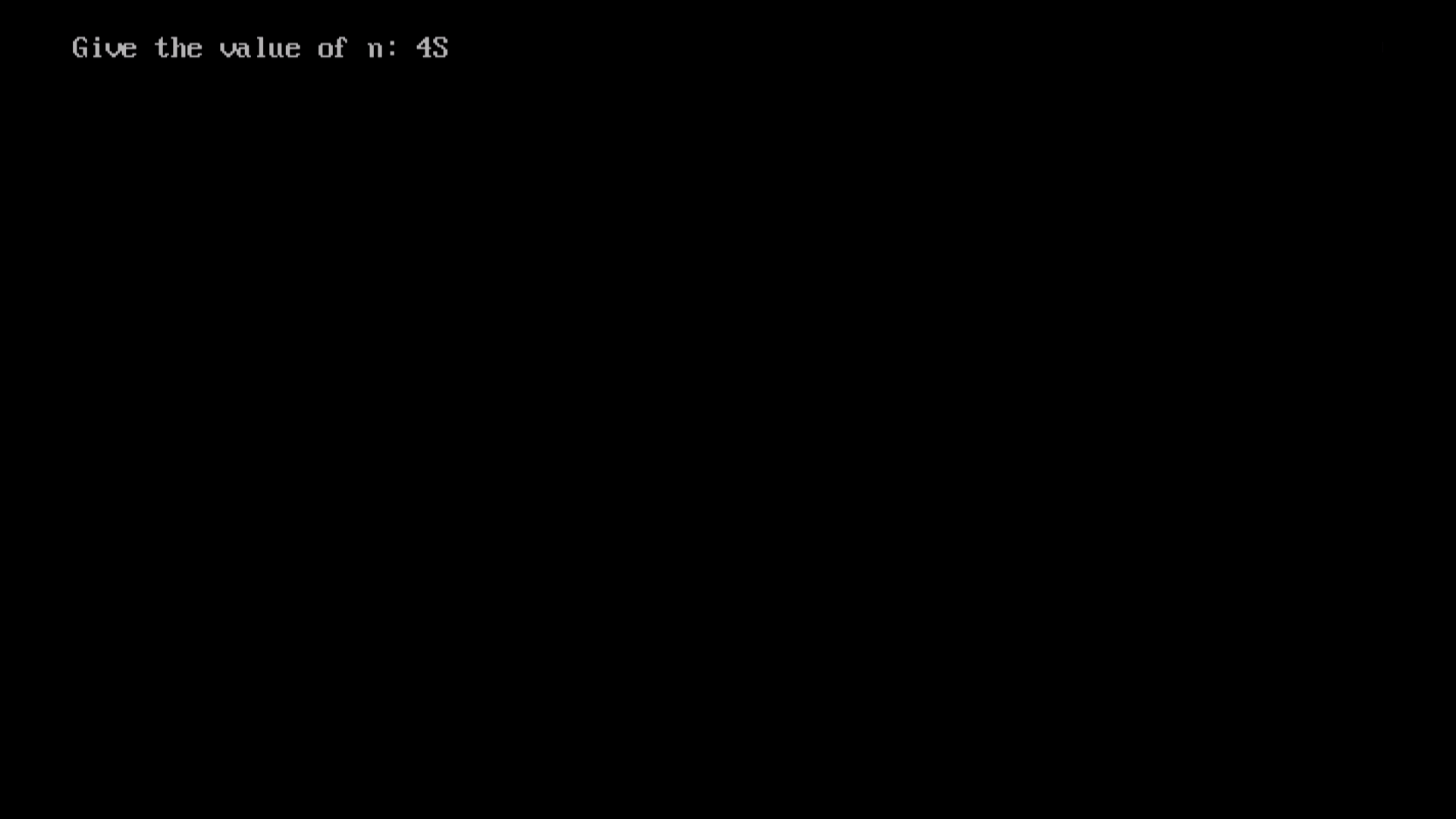
hilbert(r,d,l,u,n,h,x,y);

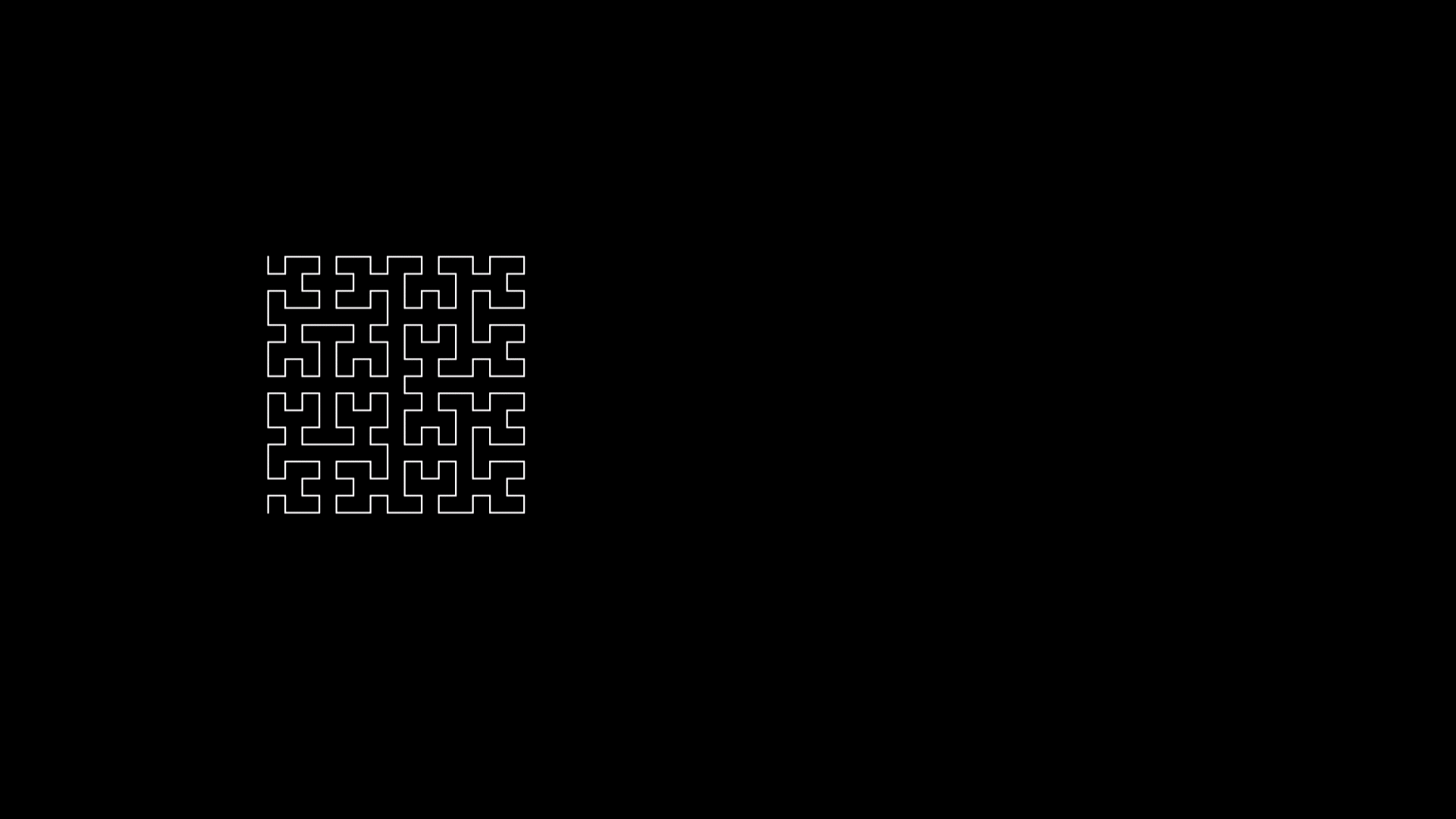
delay(10000);

closegraph();

return 0; }

**OUTPUT :**





ASSIGNMENT NO :- 6

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<iostream>

#include<stdlib.h>

#ifdef \_\_APPLE\_\_

#include<openGL/openGL.h>

#include<GLUT/glut.h>

#else

#include<GL/glut.h>

#endif

using namespace std;

float ballX = -0.8f;

float ballY = -0.3f;

float ballZ = -1.2f;

float colR=3.0;

float colG=1.5;

float colB=1.0;

float bgColR=0.0;

float bgColG=0.0;

float bgColB=0.0;

static int flag=1;

void drawBall(void) {

glColor3f(colR,colG,colB); //set ball colour

glTranslatef(ballX,ballY,ballZ); //moving it toward the screen a bit on creation

glutSolidSphere (0.05, 30, 30); //create ball.

}

void drawAv(void) {

glBegin(GL\_POLYGON);

glColor3f(1.0,1.0,1.0);

glVertex3f(-0.9,-0.7,-1.0);

glVertex3f(-0.5,-0.1,-1.0);

glVertex3f(-0.2,-1.0,-1.0);

glVertex3f(0.5,0.0,-1.0);

glVertex3f(0.6,-0.2,-1.0);

glVertex3f(0.9,-0.7,-1.0);

glEnd();

}

void drawClouds(){}

void keyPress(int key, int x, int y)

{

if(key==GLUT\_KEY\_RIGHT)

ballX -= 0.05f;

if(key==GLUT\_KEY\_LEFT)

ballX += 0.05f;2

glutPostRedisplay();

}

void initRendering() {

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING); //Enable lighting

glEnable(GL\_LIGHT0); //Enable light #0

glEnable(GL\_LIGHT1); //Enable light #1

glEnable(GL\_NORMALIZE); //Automatically normalize normals

//glShadeModel(GL\_SMOOTH); //Enable smooth shading

//Called when the window is resized

void handleResize(int w, int h) {

//Tell OpenGL how to convert from coordinates to pixel values

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION); //Switch to setting the camera perspective

//Set the camera perspective

glLoadIdentity(); //Reset the camera

gluPerspective(45.0, //The camera angle

(double)w / (double)h, //The width-to-height ratio

1.0, //The near z clipping coordinate

200.0); //The far z clipping coordinate

}

void drawScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glClearColor(bgColR,bgColG,bgColB,0.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//Add ambient light

GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //Color (0.2, 0.2, 0.2)

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambientColor);

//Add positioned light

GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Color (0.5, 0.5, 0.5)

GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f}; //Positioned at (4, 0, 8)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightColor0);

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos0);

//Add directed light

GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f}; //Color (0.5, 0.2, 0.2)

//Coming from the direction (-1, 0.5, 0.5)

GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, lightColor1);

glLightfv(GL\_LIGHT1, GL\_POSITION, lightPos1);

//drawing the SUN

glPushMatrix();

drawBall();

glPopMatrix();

//drawing the Mount Avarest

glPushMatrix();

drawAv();

glPopMatrix();

//drawing the Clouds

glPushMatrix();

drawClouds();

glPopMatrix();

glutSwapBuffers();

}

//float \_angle = 30.0f;

void update(int value) {

if(ballX>0.9f)

{

ballX = -0.8f;

ballY = -0.3f;

flag=1;

colR=2.0;

colG=1.50;

colB=1.0;

bgColB=0.0;

}

if(flag)

{

ballX += 0.001f;

ballY +=0.0007f;

colR-=0.001;

//colG+=0.002;

colB+=0.005;

bgColB+=0.001;

if(ballX>0.01)

{

flag=0;

}

}

if (!flag)

{

ballX += 0.001f;

ballY -=0.0007f;

colR+=0.001;

colB-=0.01;

bgColB-=0.001;

if(ballX<-0.3)

{

flag=1;

}

}

glutPostRedisplay(); //Tell GLUT that the display has changed

//Tell GLUT to call update again in 25 milliseconds

glutTimerFunc(25, update, 0);

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(400,400);

glutCreateWindow("Sun");

initRendering();

glutDisplayFunc(drawScene);

glutFullScreen();

glutSpecialFunc(keyPress);

glutReshapeFunc(handleResize);

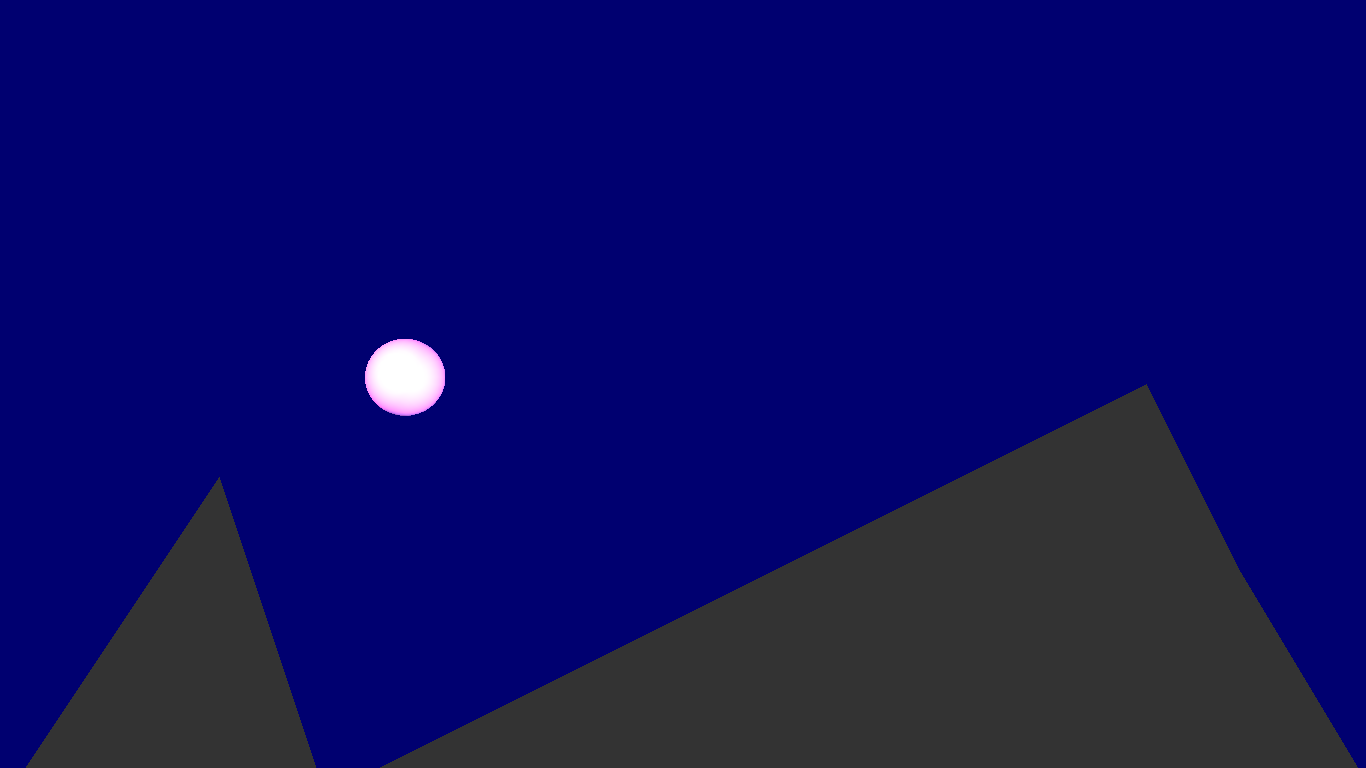
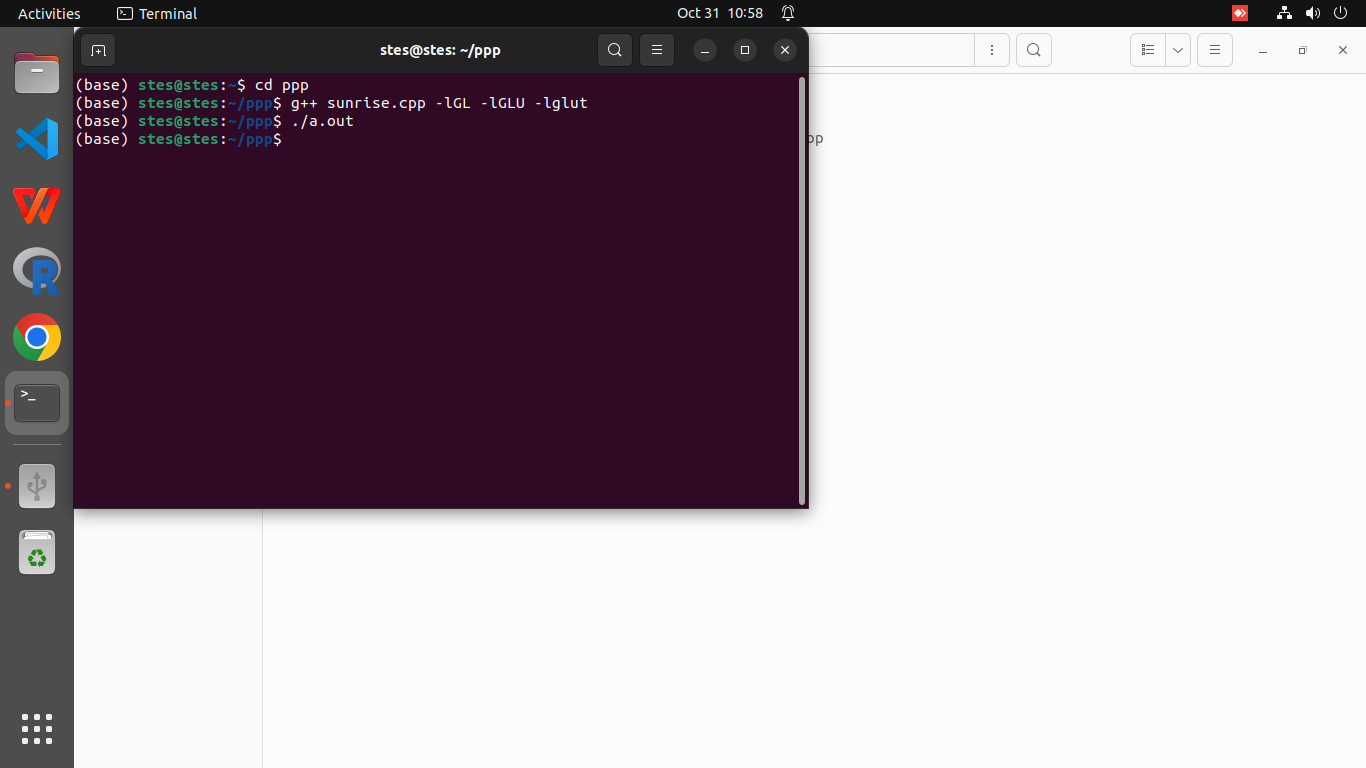
glutTimerFunc(25, update, 0);

glutMainLoop();

return(0);

}

**OUTPUT :**



ASSIGNMENT NO :- 7

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

#include<dos.h>

#include<iostream.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"//TURBOC3//BGI");

float x=1,y=0.00000,j=0.5,count=0.1;

float r=15;

setcolor(14);

line(0,215,650,215);

for(int k=0;k<=7;k++)

{

for(float i=90;i<270;i+=10)

{

y=cos(((i\*22/7)/180))/j;

if(y>0)

y=-y;

x+=5;

setcolor(14);

setfillstyle(1,14);

circle(x,y\*100+200,r);

floodfill(x,y\*100+200,14);

delay(100);

setcolor(0);

setfillstyle(1,0);

circle(x,y\*100+200,r);

floodfill(x,y\*100+200,0);

}

j+=count;

count+=0.1;

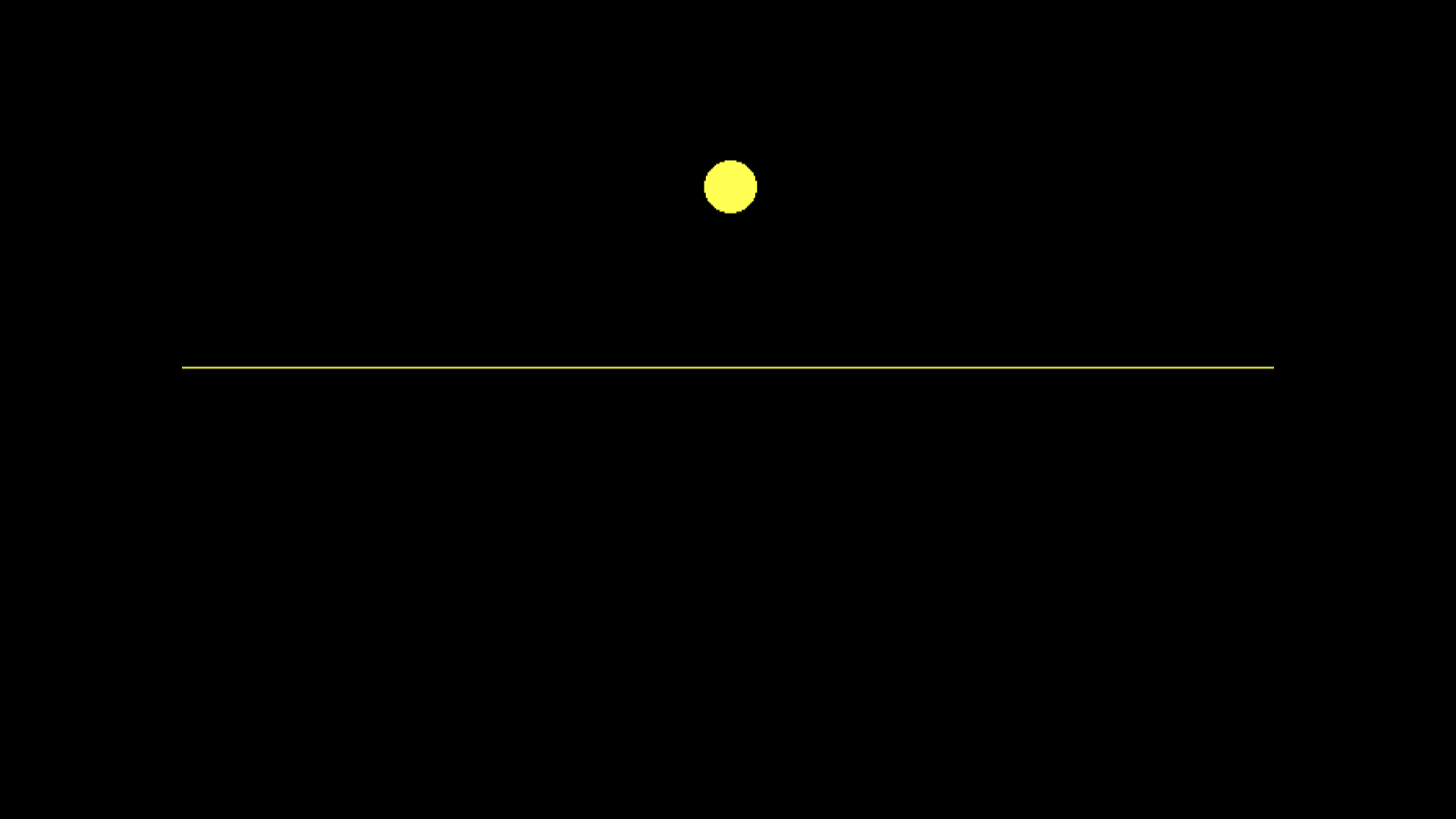
}

getch();

return 0;

}

**OUTPUT :**



ASSIGNMENT NO :- MINI PROJECT

NAME : ARYAN SACHIN SHITOLE

CLASS : SE – 3 DIV : B

ROLL NO : 205C039

PROGRAMMING CODE :

// C++ program to draw the moving

// cycle using computer graphics

#include <conio.h>

#include <dos.h>

#include <graphics.h>

#include <iostream.h>

// Driver code

int main()

{

int gd = DETECT, gm, i, a;

// Path of the program

initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

// Move the cycle

for (i = 0; i < 600; i++) {

// Upper body of cycle

line(50 + i, 405, 100 + i, 405);

line(75 + i, 375, 125 + i, 375);

line(50 + i, 405, 75 + i, 375);

line(100 + i, 405, 100 + i, 345);

line(150 + i, 405, 100 + i, 345);

line(75 + i, 345, 75 + i, 370);

line(70 + i, 370, 80 + i, 370);

line(80 + i, 345, 100 + i, 345);

// Wheel

circle(150 + i, 405, 30);

circle(50 + i, 405, 30);

// Road

line(0, 436, getmaxx(), 436);

// Stone

rectangle(getmaxx() - i, 436,

650 - i, 431);

// Stop the screen for 10 secs

delay(10);

// Clear the screen

cleardevice();

}

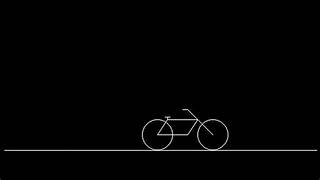
getch();

// Close the graph

closegraph();

}

**OUTPUT :**

****