**PROGRAM 4**

Name:- Kaustubh S Kabra

Class:-Second Year Engineering Comp-1

Roll No:- 20

**Write a C++ program to draw a 2D object and perform the following transformations:- 1. Scaling 2. Translation 3. Rotation**

Program:-

#include<iostream.h>

#include<math.h>

#include<graphics.h>

#include<conio.h>

class translation

{

int val;

public:

void setvalue(int temp)

{

val=temp;

}

int disp()

{

return(val);

}

translation operator+(translation o)

{

translation t;

t.val=val+o.val;

return(t);

}

};

class scale

{

int val;

public:

void setvalue(int temp){ val=temp;

}

int disp()

{

return(val);

}

scale operator\*(scale o)

{

scale s;

s.val=val\*o.val;

return(s);

}

};

class rotation

{

public:

float b[3][3],a[3][3];

int x1,y1,x2,y2,x3,y3;

rotation()

{

x1=100,y1=100,x2=300,y2=100,x3=150,y3=50;

}

rotation ret()

{

rotation t1;

t1.b[0][0]=x1;

t1.b[0][1]=y1;

t1.b[0][2]=0;

t1.b[1][0]=x2;

t1.b[1][1]=y2;

t1.b[1][2]=0;

t1.b[2][0]=x3;

t1.b[2][1]=y3;

t1.b[2][2]=1;

return(t1);

}

rotation ret1()

{

rotation t2;

float t;

float a1,a2,a3;

t=45;

t=t\*3.14/180;

a1=cos(t);

a2=sin(t);

a3=-sin(t);

t2.a[0][0]=a1;

t2.a[0][1]=a2;

t2.a[0][2]=0;

t2.a[1][0]=a3;

t2.a[1][1]=a1;

t2.a[1][2]=0;

t2.a[2][0]=0;

t2.a[2][1]=0;

t2.a[2][2]=1;

return(t2);

}

rotation operator\*(rotation o)

{

rotation t;

t.b[0][0]=((b[0][0]\*o.a[0][0])+(b[0][1]\*o.a[1][0])+(b[0][2]\*o.a[2][0]));

t.b[0][1]=((b[0][0]\*o.a[0][1])+(b[0][1]\*o.a[1][1])+(b[0][2]\*o.a[2][1]));

t.b[0][2]=((b[0][0]\*o.a[0][2])+(b[0][1]\*o.a[1][2])+(b[0][2]\*o.a[2][2]));

t.b[1][0]=((b[1][0]\*o.a[0][0])+(b[1][1]\*o.a[1][0])+(b[1][2]\*o.a[2][0]));

t.b[1][1]=((b[1][0]\*o.a[0][1])+(b[1][1]\*o.a[1][1])+(b[1][2]\*o.a[2][1]));

t.b[1][2]=((b[1][0]\*o.a[0][2])+(b[1][1]\*o.a[1][2])+(b[1][2]\*o.a[2][2]));

t.b[2][0]=((b[2][0]\*o.a[0][0])+(b[2][1]\*o.a[1][0])+(b[2][2]\*o.a[2][0]));

t.b[2][1]=((b[2][0]\*o.a[0][0])+(b[2][1]\*o.a[1][0])+(b[2][2]\*o.a[2][0]));

t.b[2][2]=((b[2][0]\*o.a[0][0])+(b[2][1]\*o.a[1][0])+(b[2][2]\*o.a[2][0]));

return(t);

}

};

int main()

{

int gd=DETECT,gm=0;

int ch,flag=0;

int x1=100,y1=100,x2=300,y2=100,x3=150,y3=50,tx=50,ty=50; int

a1=100,b1=100,a2=300,b2=100,a3=150,b3=50,sx=2,sy=3;

translation t1,t2,t3,t4,t5,t6,t7,t8;

rotation r1,r2,r3,r4;

scale s1,s2,s3,s4,s5,s6,s7,s8;

do

{

cout<<"\n\t\t MENU";

cout<<"\n 1.Translation \n 2.Scaling \n 3.Rotation\n 4.Exit \n Please Enter Your Choice:";

cin>>ch;

switch(ch)

{

case 1:

{

initgraph(&gd,&gm,"C://turboc3//bgi");

t1.setvalue(x1);

t2.setvalue(y1);

t3.setvalue(x2);

t4.setvalue(y2);

t5.setvalue(x3);

t6.setvalue(y3);

line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

line(x1,y1,x3,y3);

t7.setvalue(tx);

t8.setvalue(ty);

setcolor(GREEN);

t1=t1+t7;

t2=t2+t8;

t3=t3+t7;

t4=t4+t8;

t5=t5+t7;

t6=t6+t8;

line(t1.disp(),t2.disp(),t3.disp(),t4.disp());

line(t3.disp(),t4.disp(),t5.disp(),t6.disp());

line(t1.disp(),t2.disp(),t5.disp(),t6.disp());

break;

}

case 2:

{

initgraph(&gd,&gm,"C://turboc3//bgi");

s1.setvalue(a1);

s2.setvalue(b1);

s3.setvalue(a2);

s4.setvalue(b2);

s5.setvalue(a3);

s6.setvalue(b3);

line(a1,b1,a2,b2);

line(a2,b2,a3,b3);

line(a1,b1,a3,b3);

s7.setvalue(sx); s8.setvalue(sy);

setcolor(GREEN);

s1=s1\*s7;

s2=s2\*s8;

s3=s3\*s7;

s4=s4\*s8;

s5=s5\*s7;

s6=s6\*s8;

line(s1.disp(),s2.disp(),s3.disp(),s4.disp());

line(s3.disp(),s4.disp(),s5.disp(),s6.disp());

line(s1.disp(),s2.disp(),s5.disp(),s6.disp());

break;

}

case 3:

{

initgraph(&gd,&gm,"C://turboc3//bgi");

r2=r1.ret();

r3=r1.ret1();

int nx1,ny1,nx2,ny2,nx3,ny3; r4=r2\*r3;

nx1=r4.b[0][0];

ny1=r4.b[0][1];

nx2=r4.b[1][0];

ny2=r4.b[1][1];

nx3=r4.b[2][0];

ny3=r4.b[2][1];

line(nx1,ny1,nx2,ny2); line(nx2,ny2,nx3,ny3); line(nx3,ny3,nx1,ny1);

break;

}

case 4:

{

flag=1;

cout<<"\n\t Thank YOU";

break;

}

default:

cout<<"\n\t INVALID";

break;

}

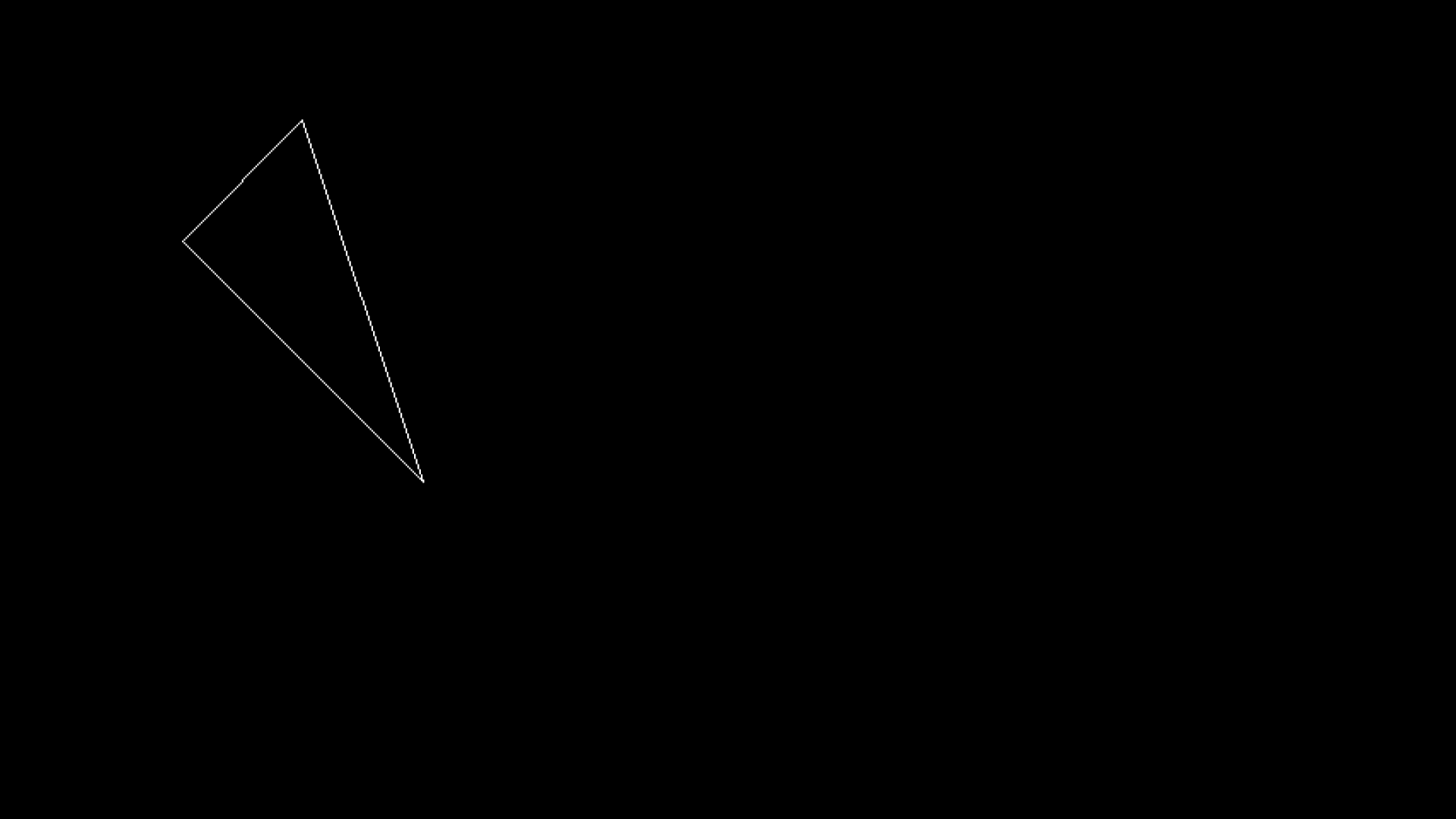
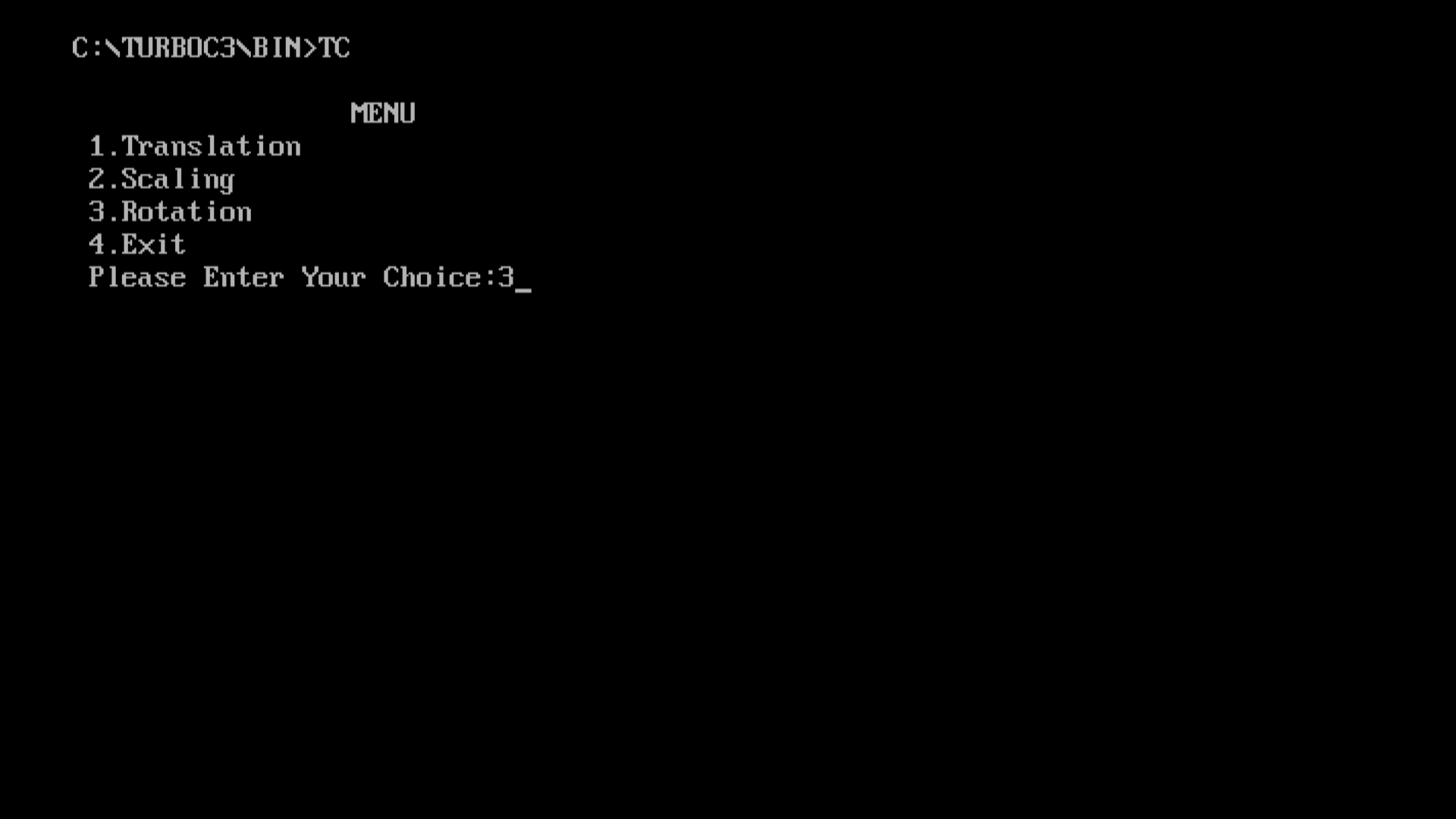
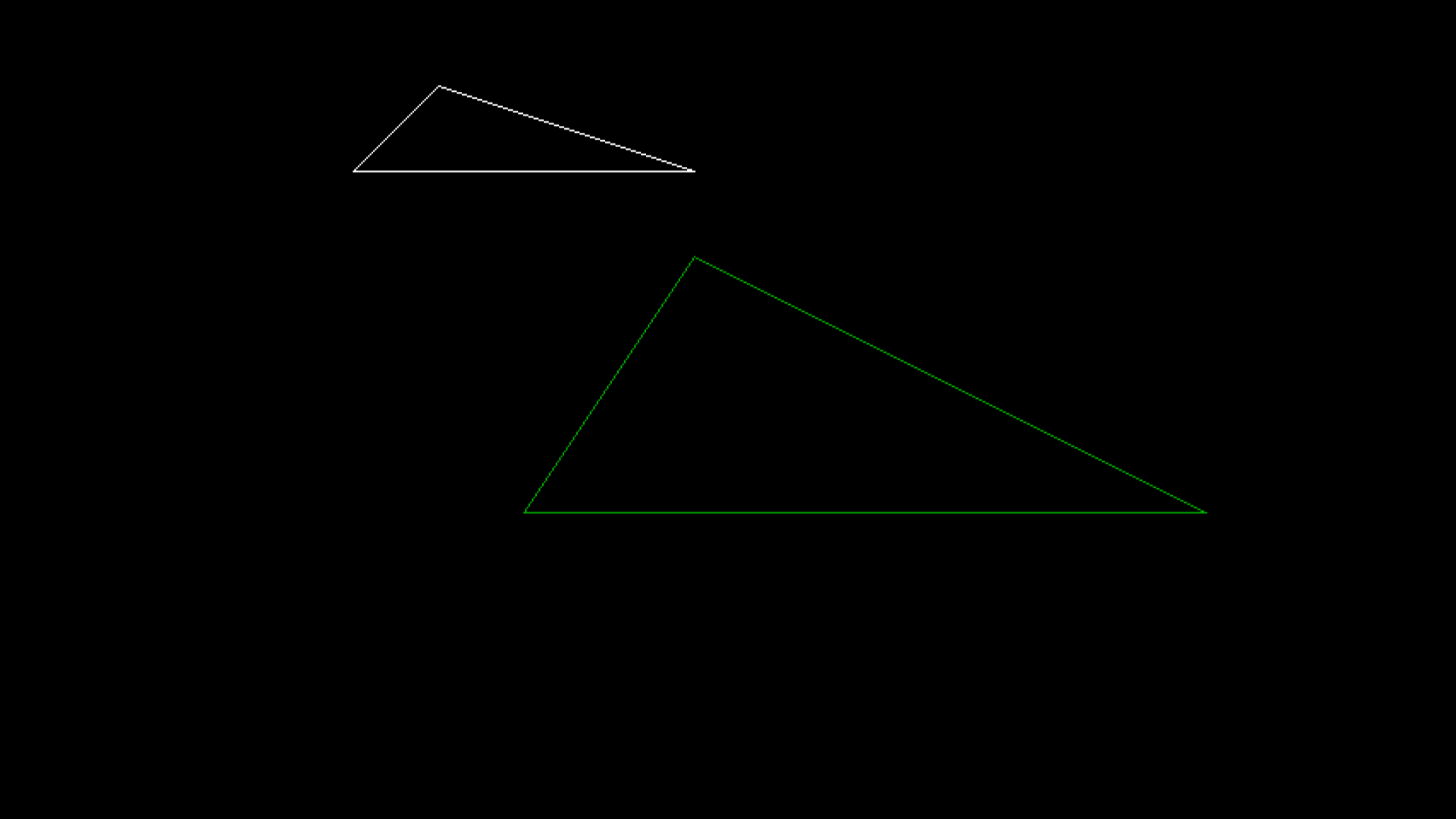
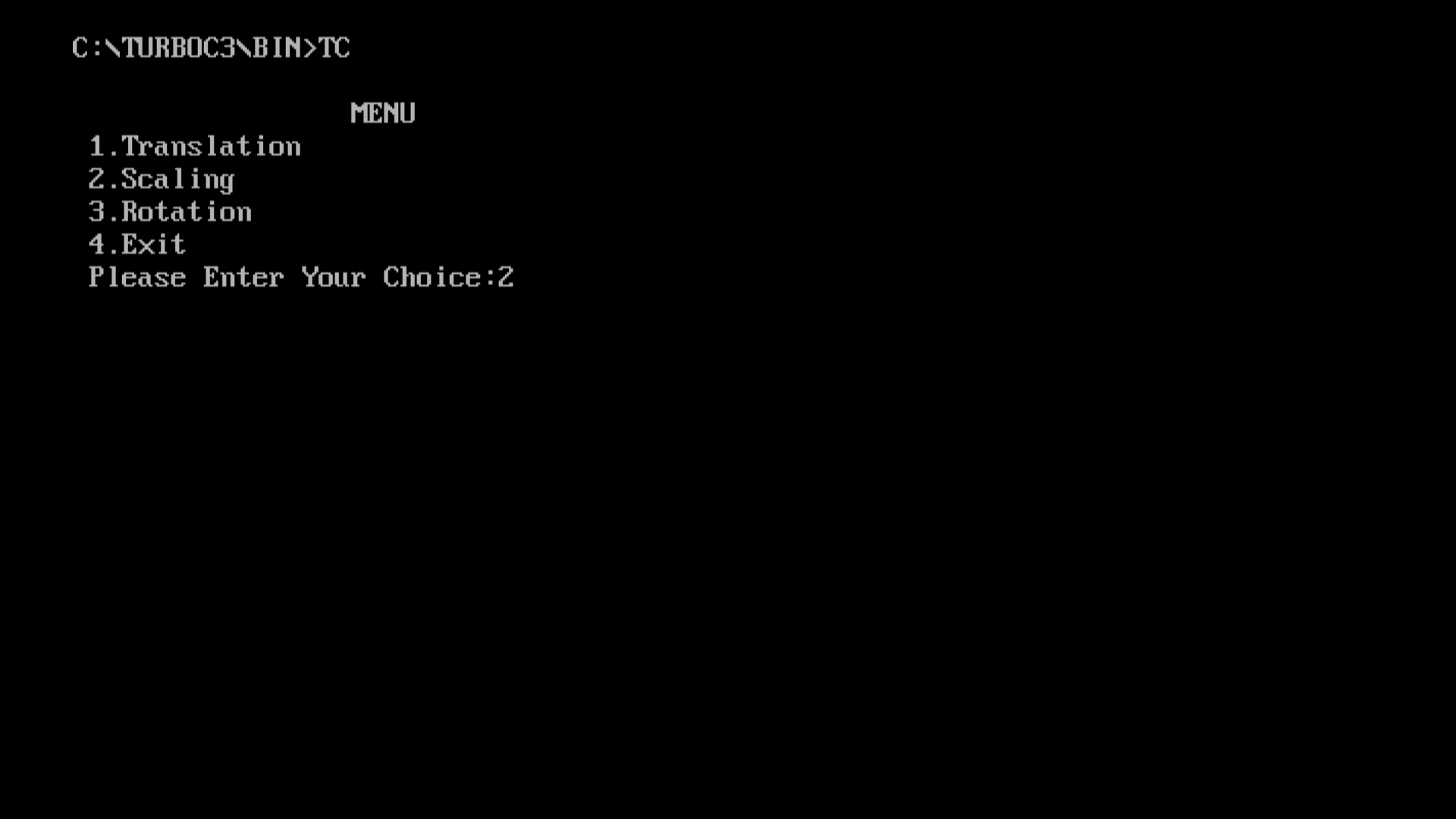
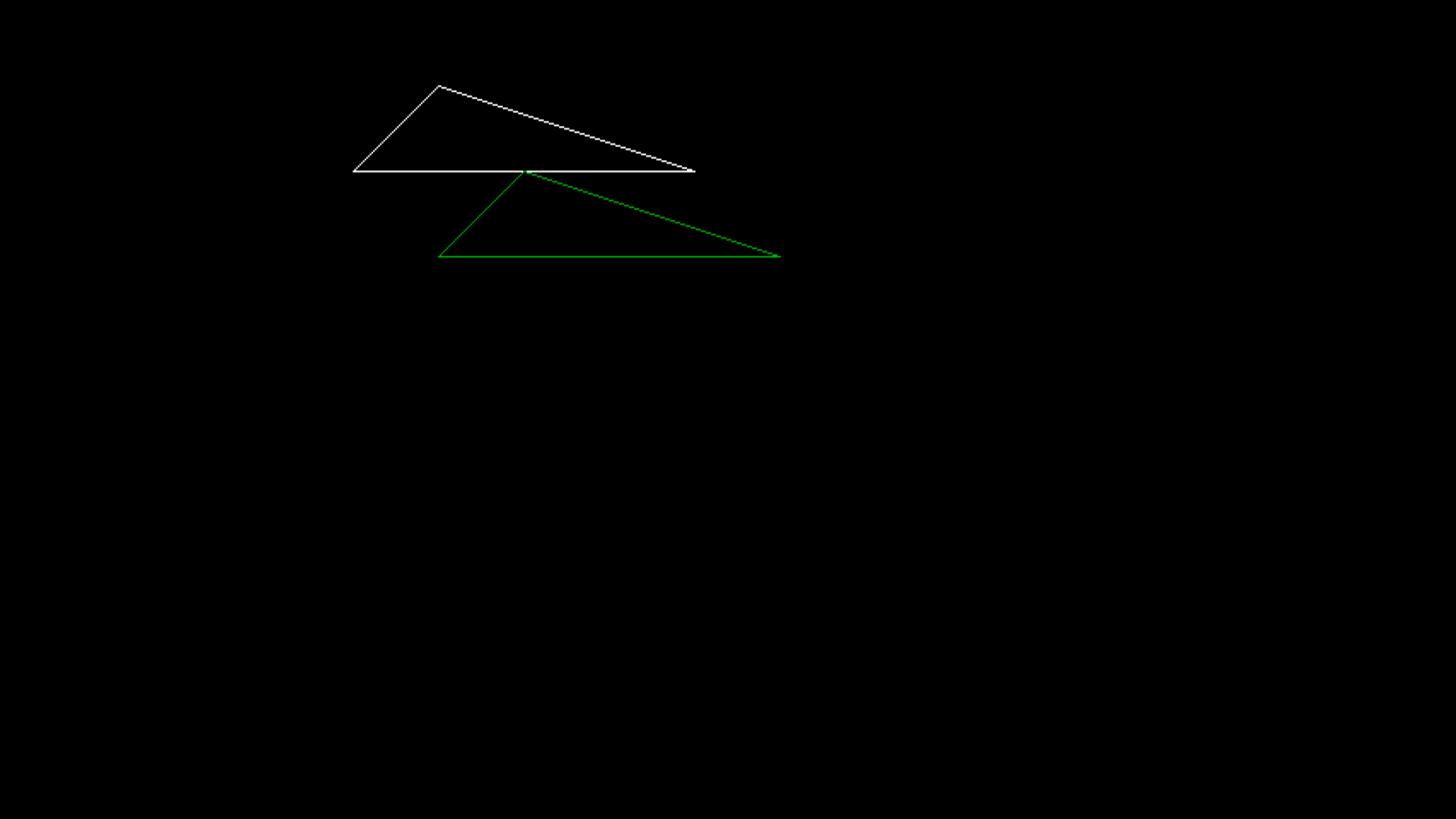
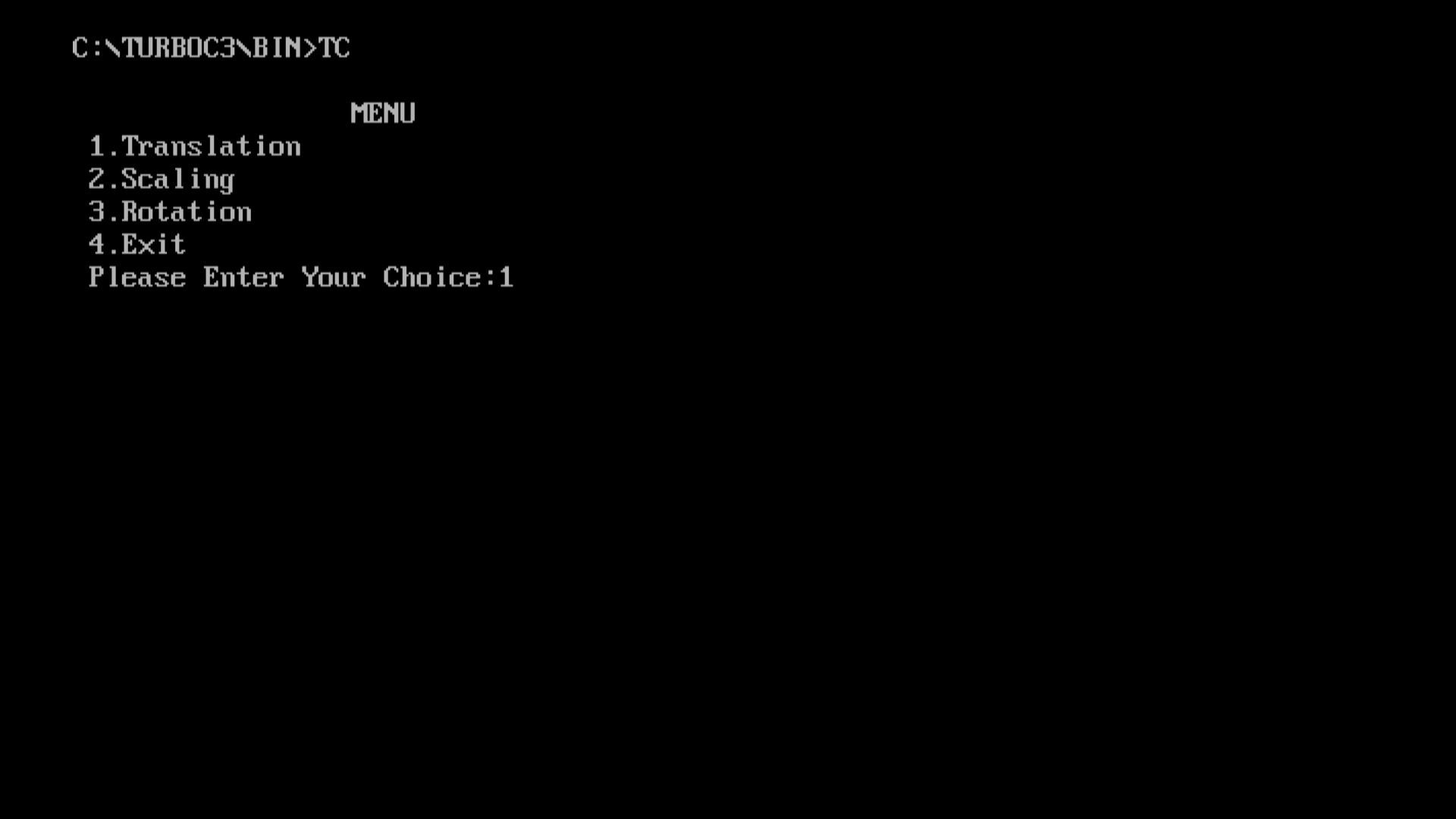
getch();

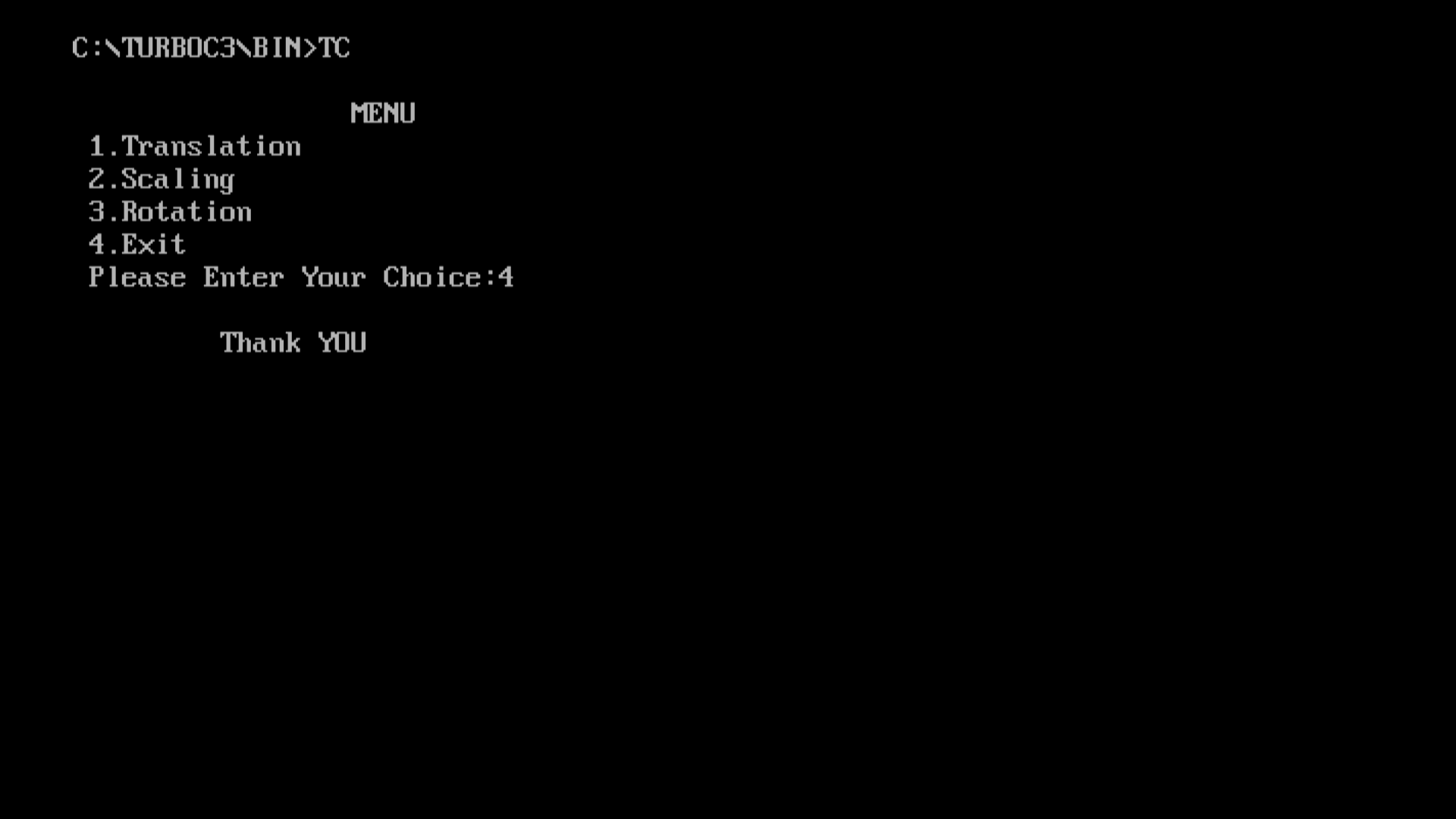
closegraph();

}while(flag==0);

return 0;

}

Output:- 



**PROGRAM 5**

Name:- Kaustubh S Kabra

Class:-Second Year Engineering Comp-1

Roll No:- 20

**Write a C++ program to generate Hilbert curve using concepts of fractals.**

Program:-

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<stdlib.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=20,y0=50,x,y,h=10,r=2,d=3,l=4,u=1;

cout<<endl<<"Enter n: ";

cin>>n;

x=x0;

y=y0;

int gd=DETECT,gm;

initgraph(&gd,&gm,"C://turboc3//bgi");

moveto(x,y);

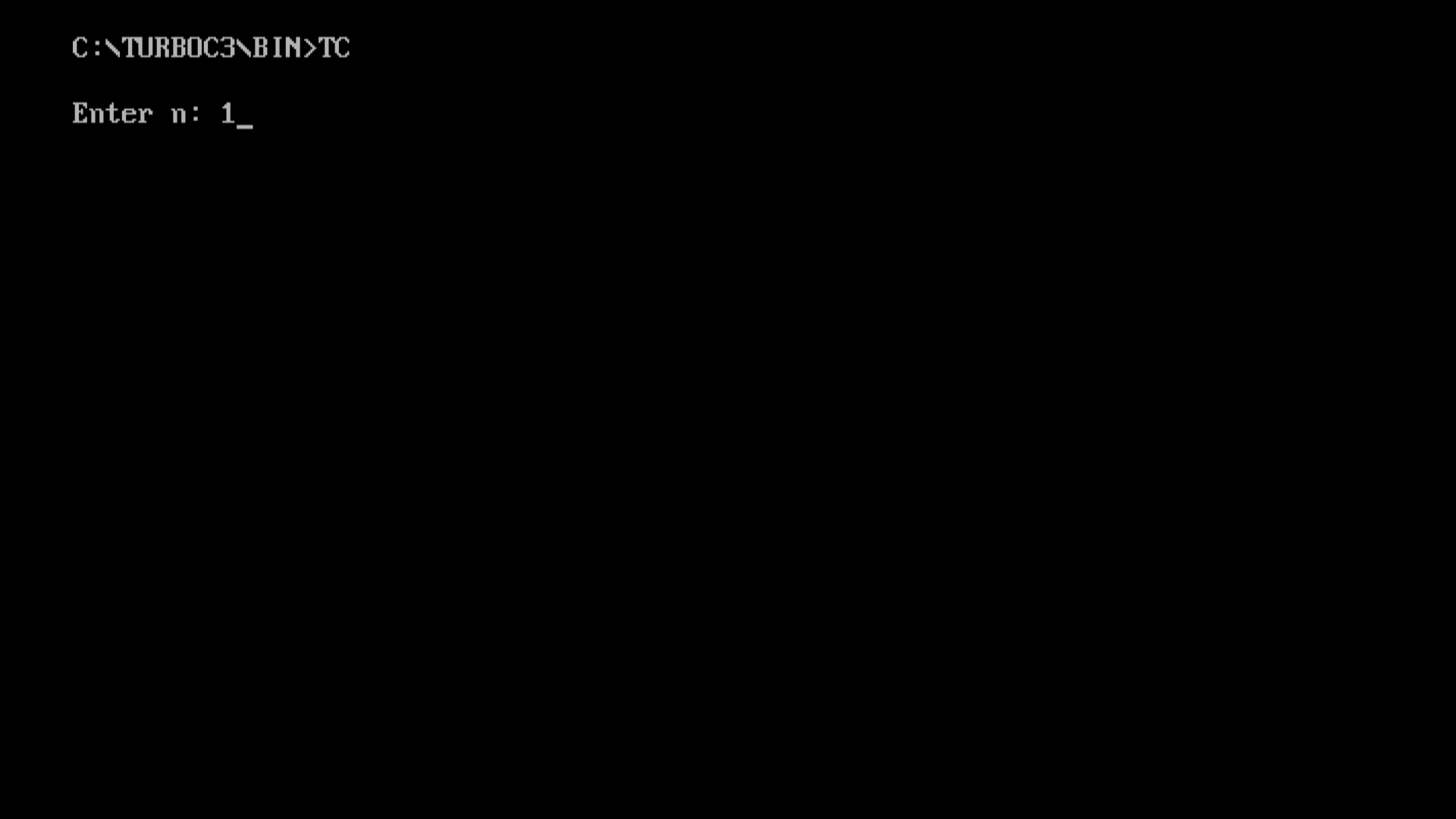
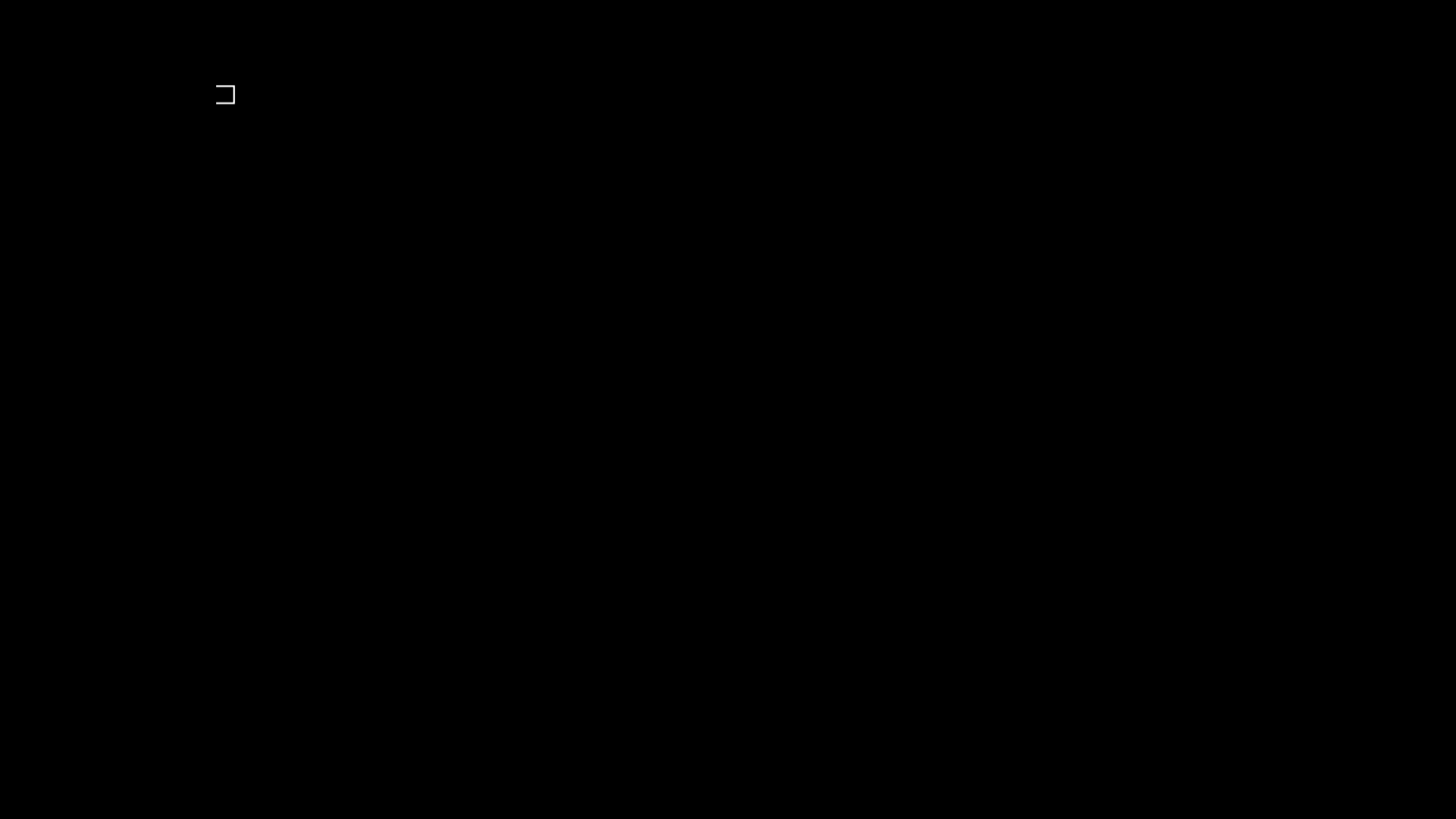
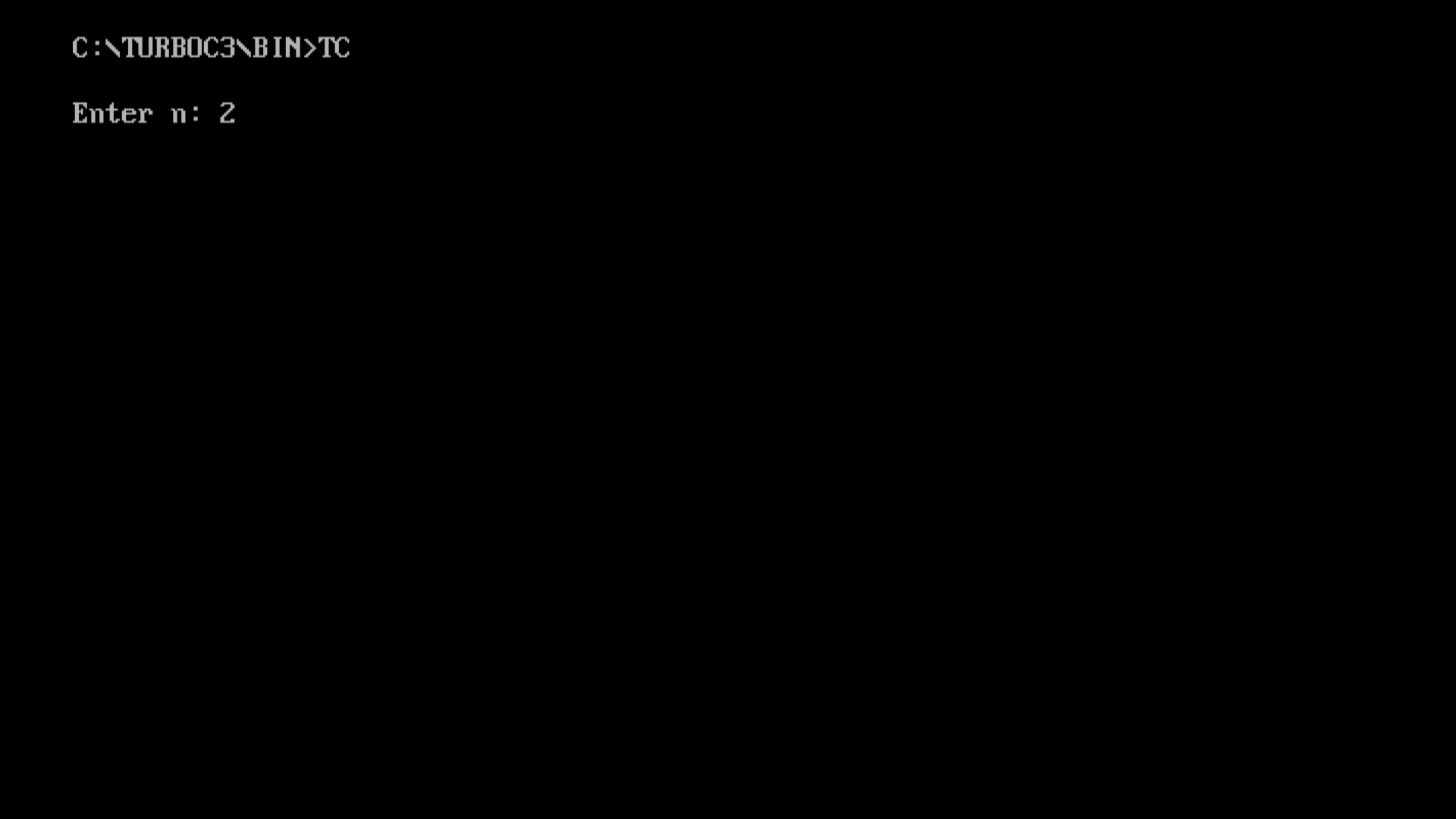
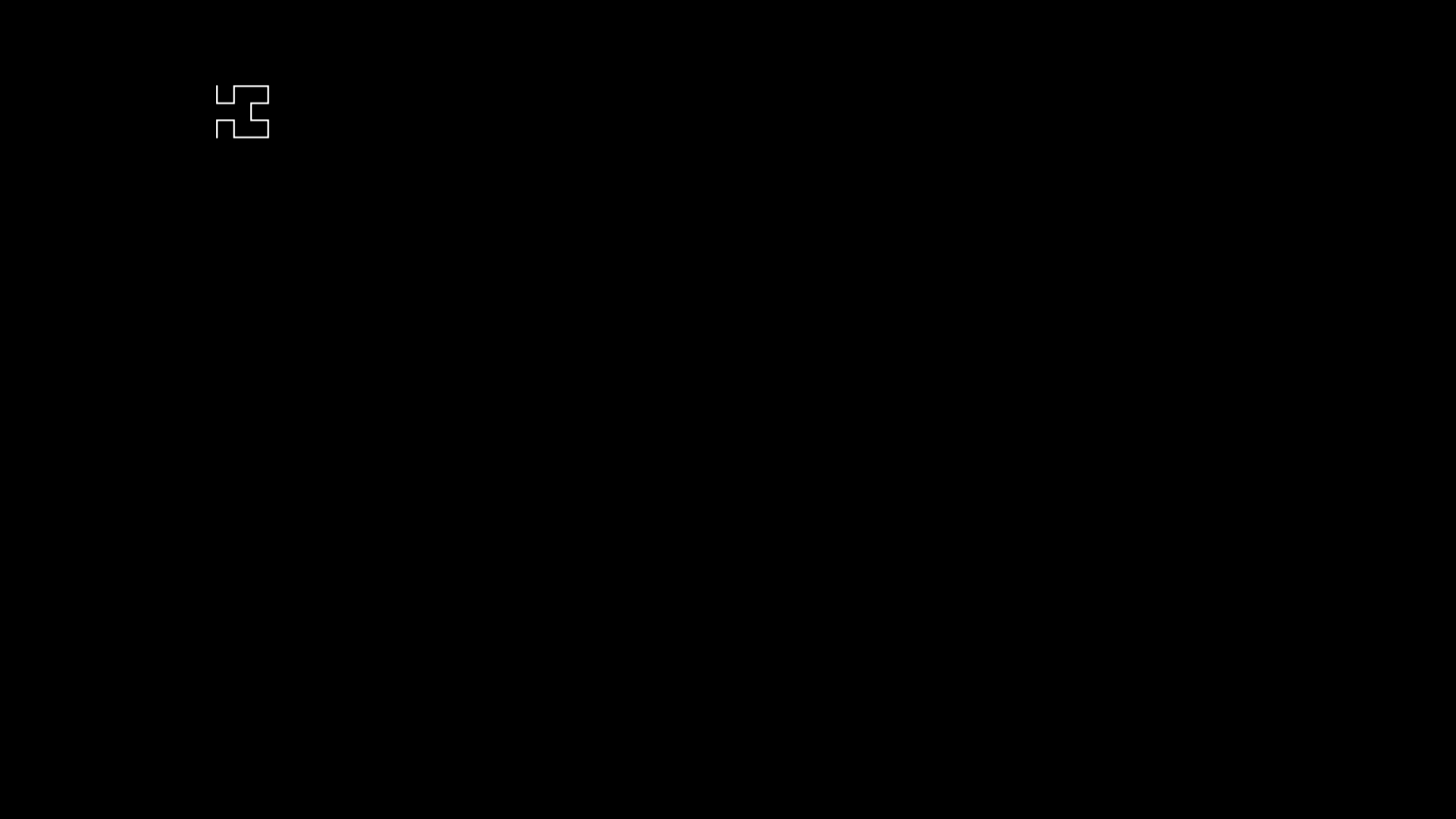
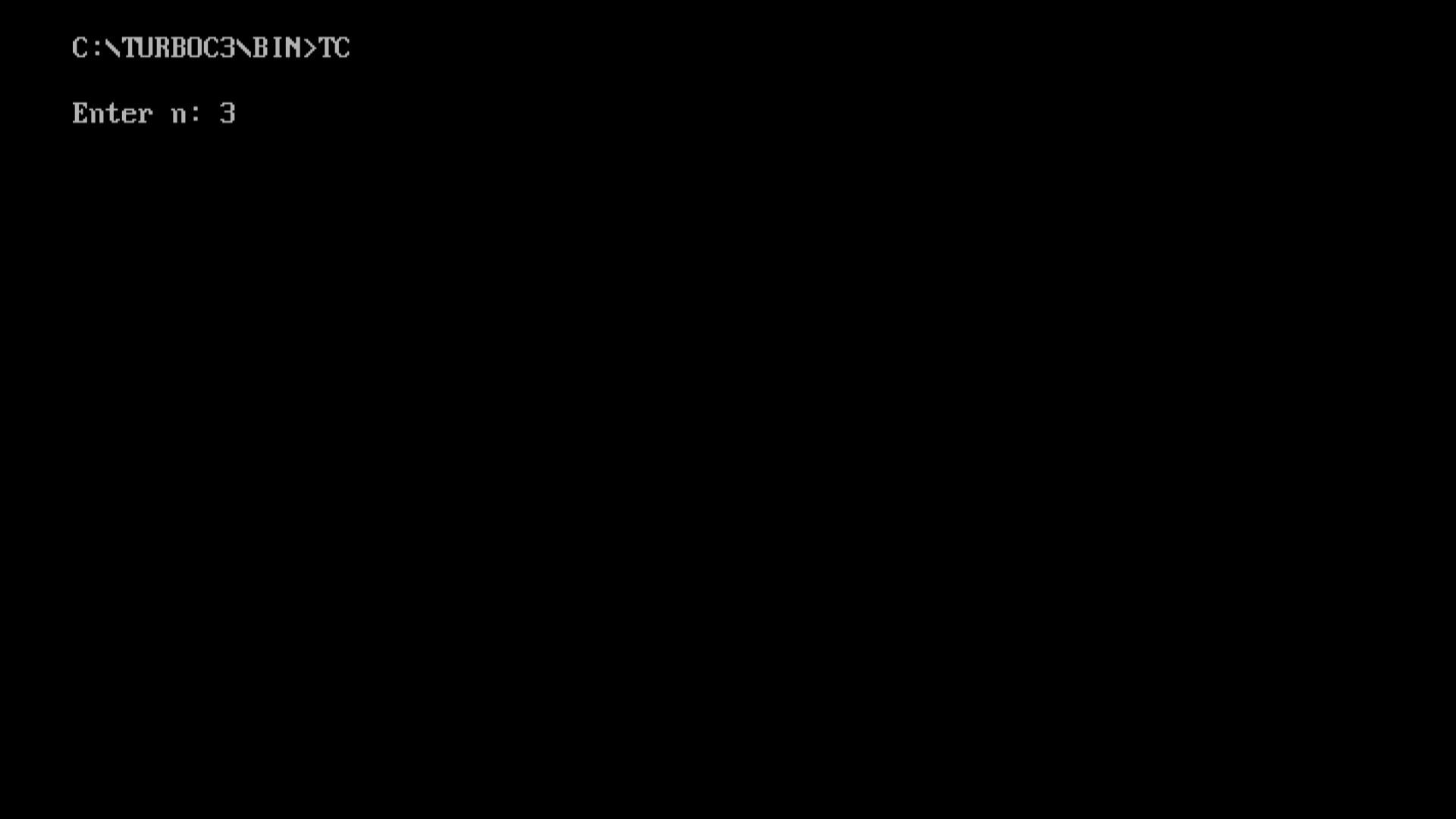
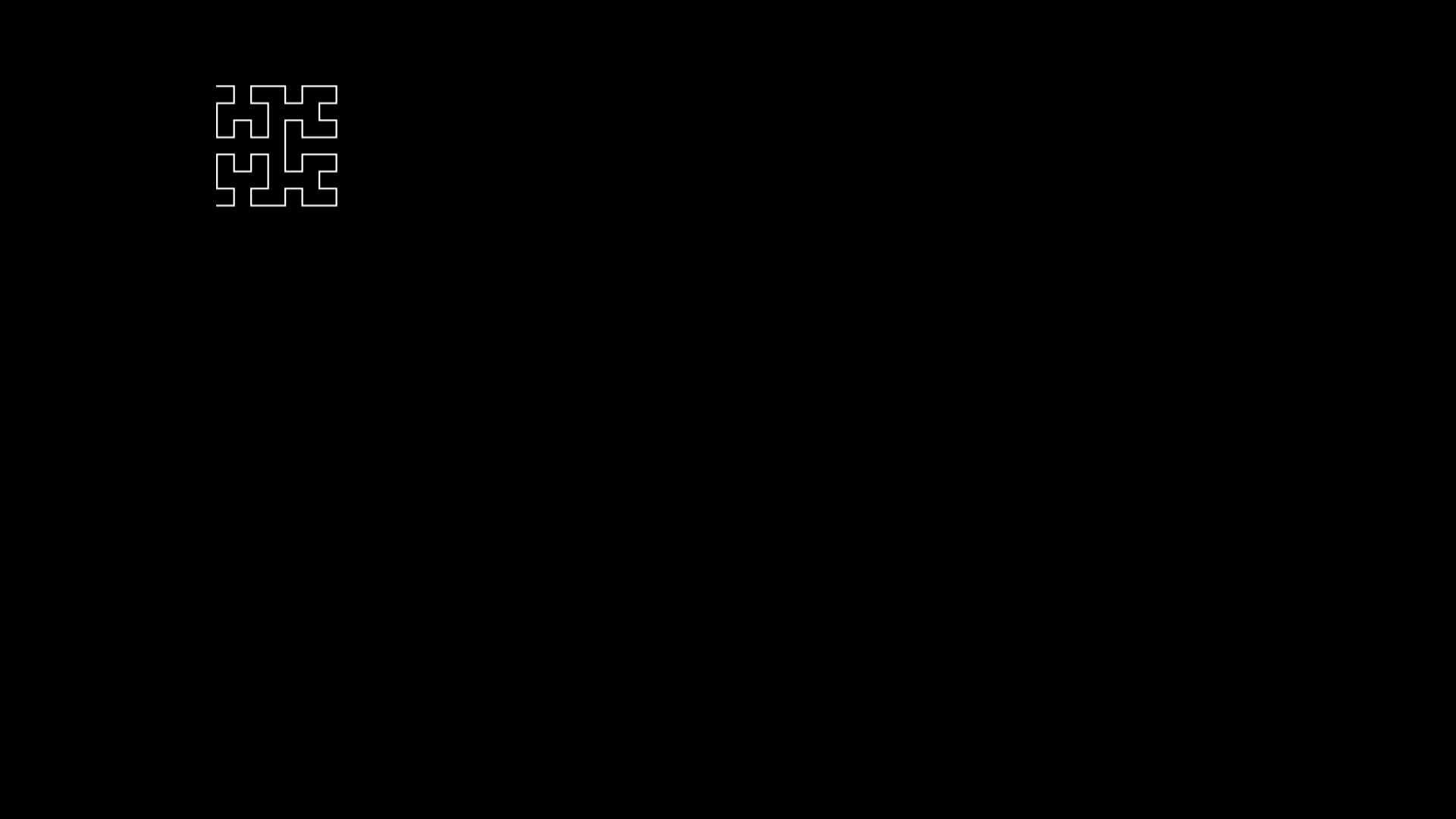
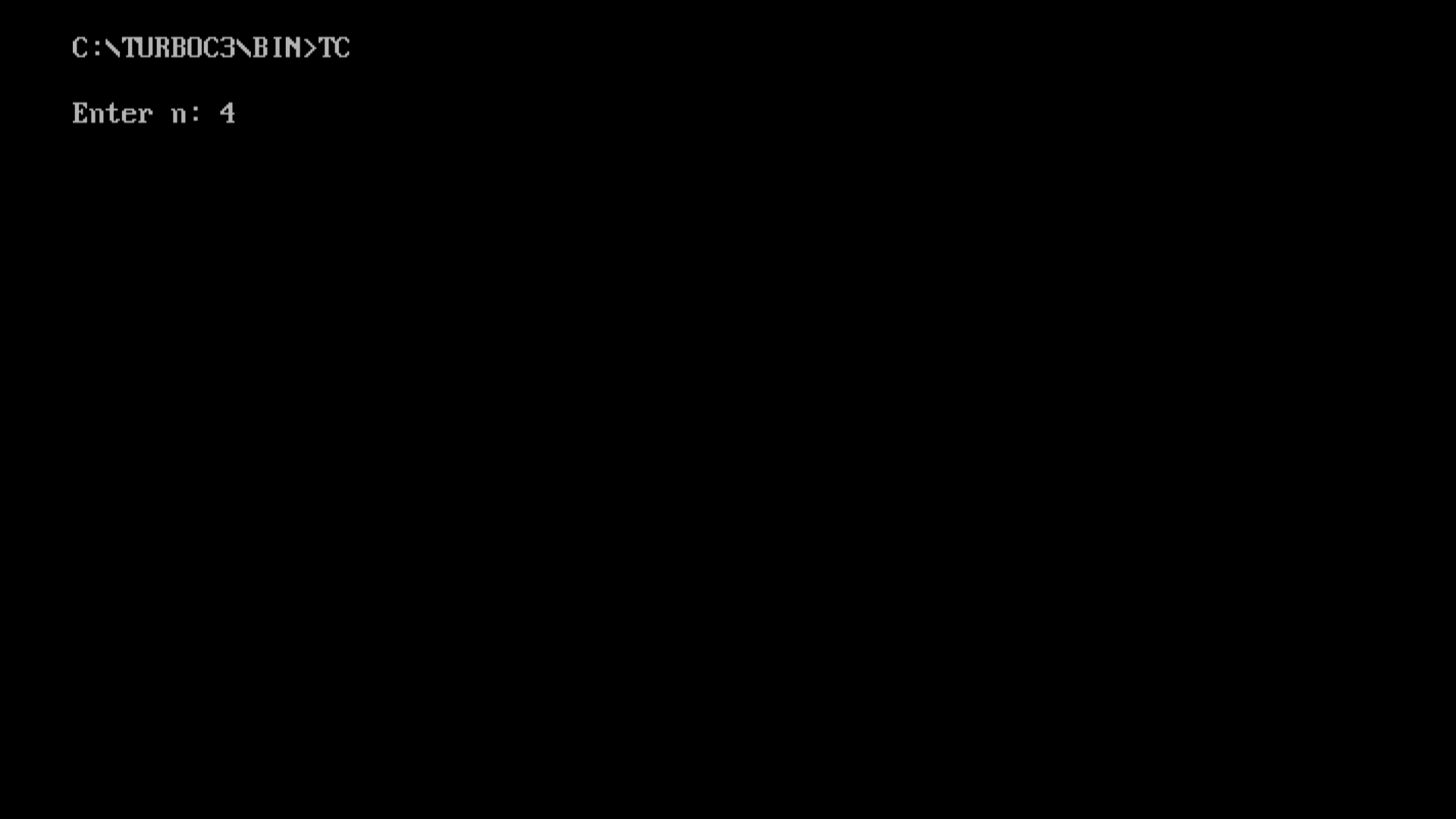
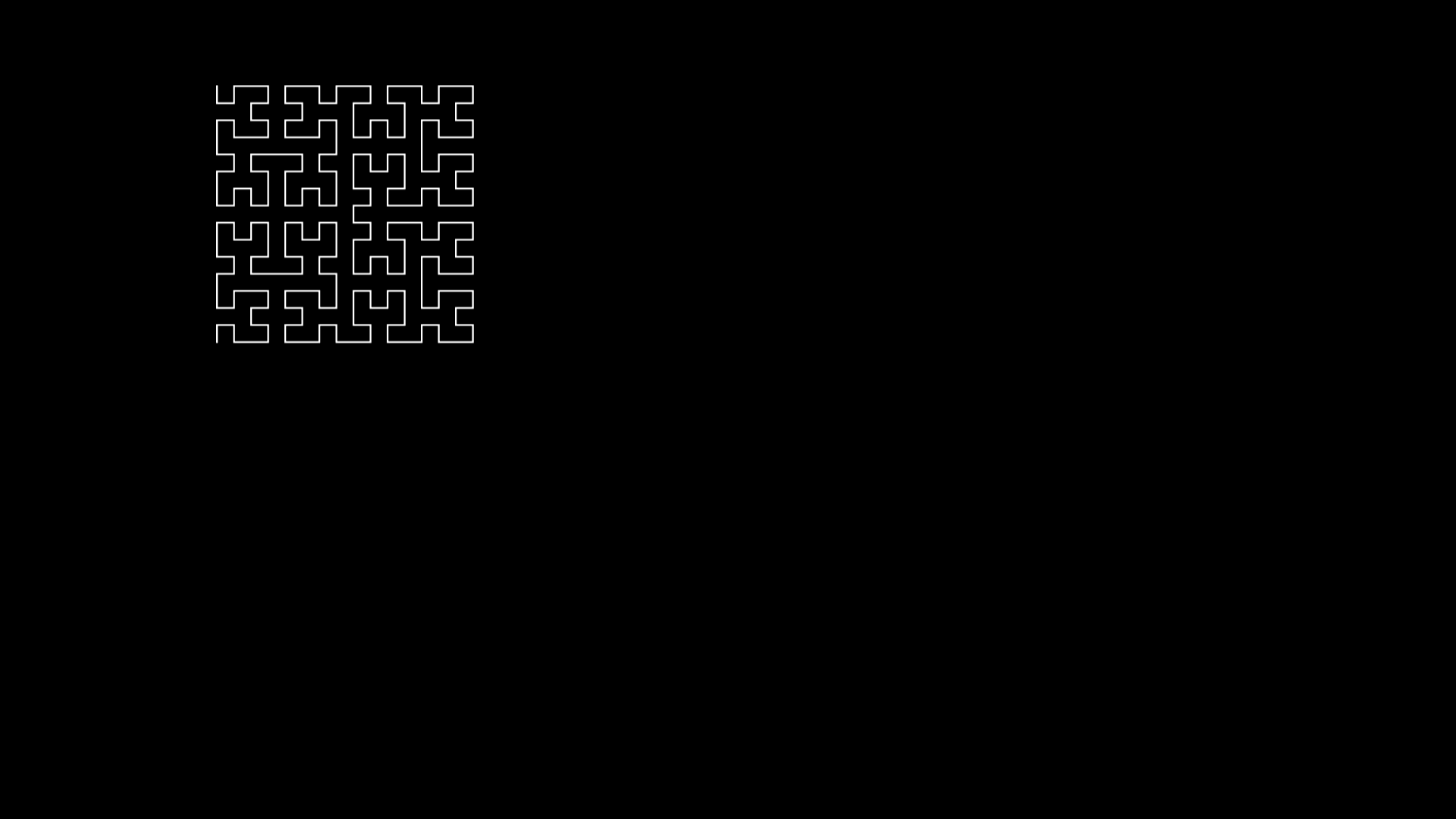
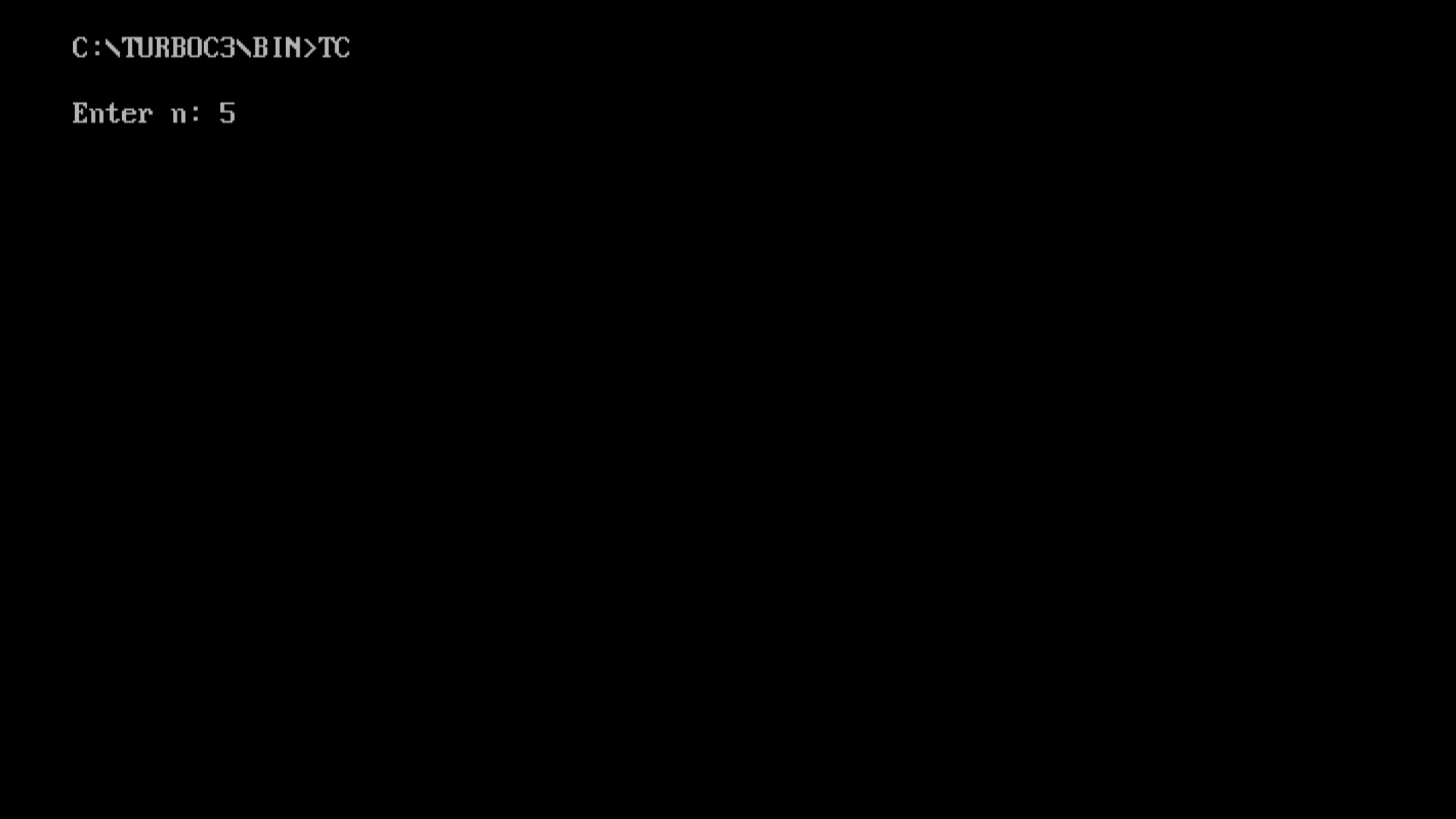
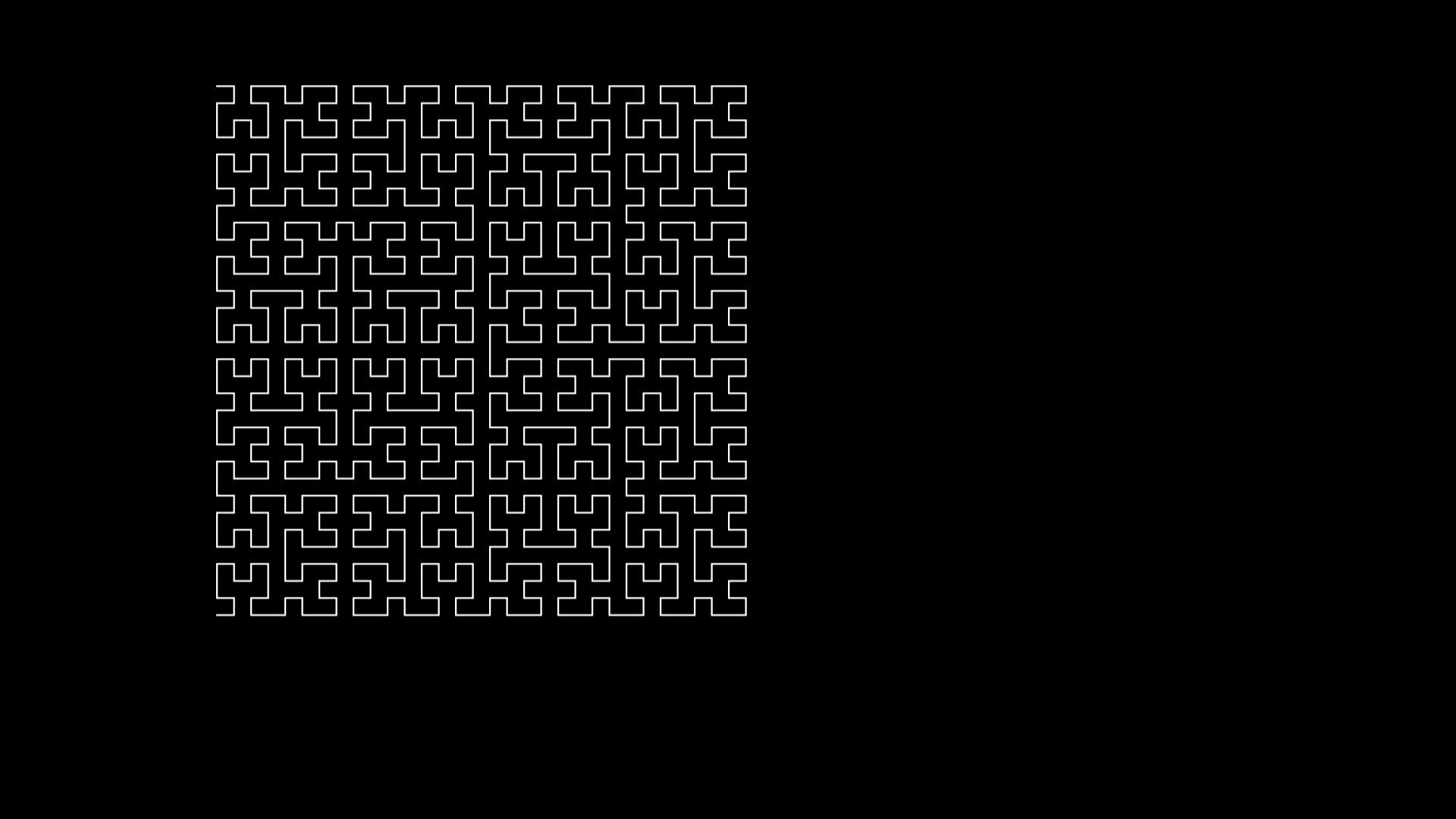
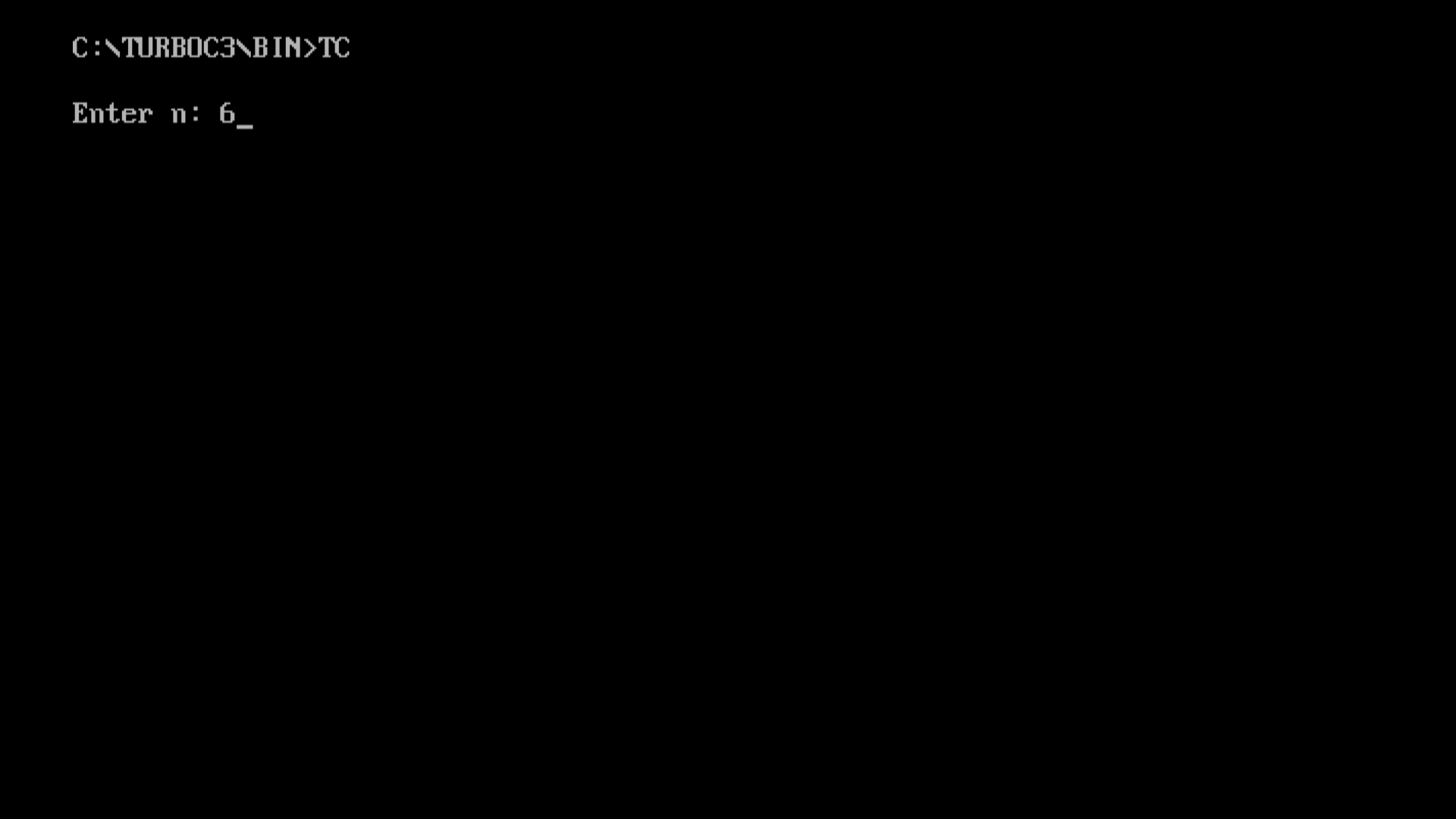
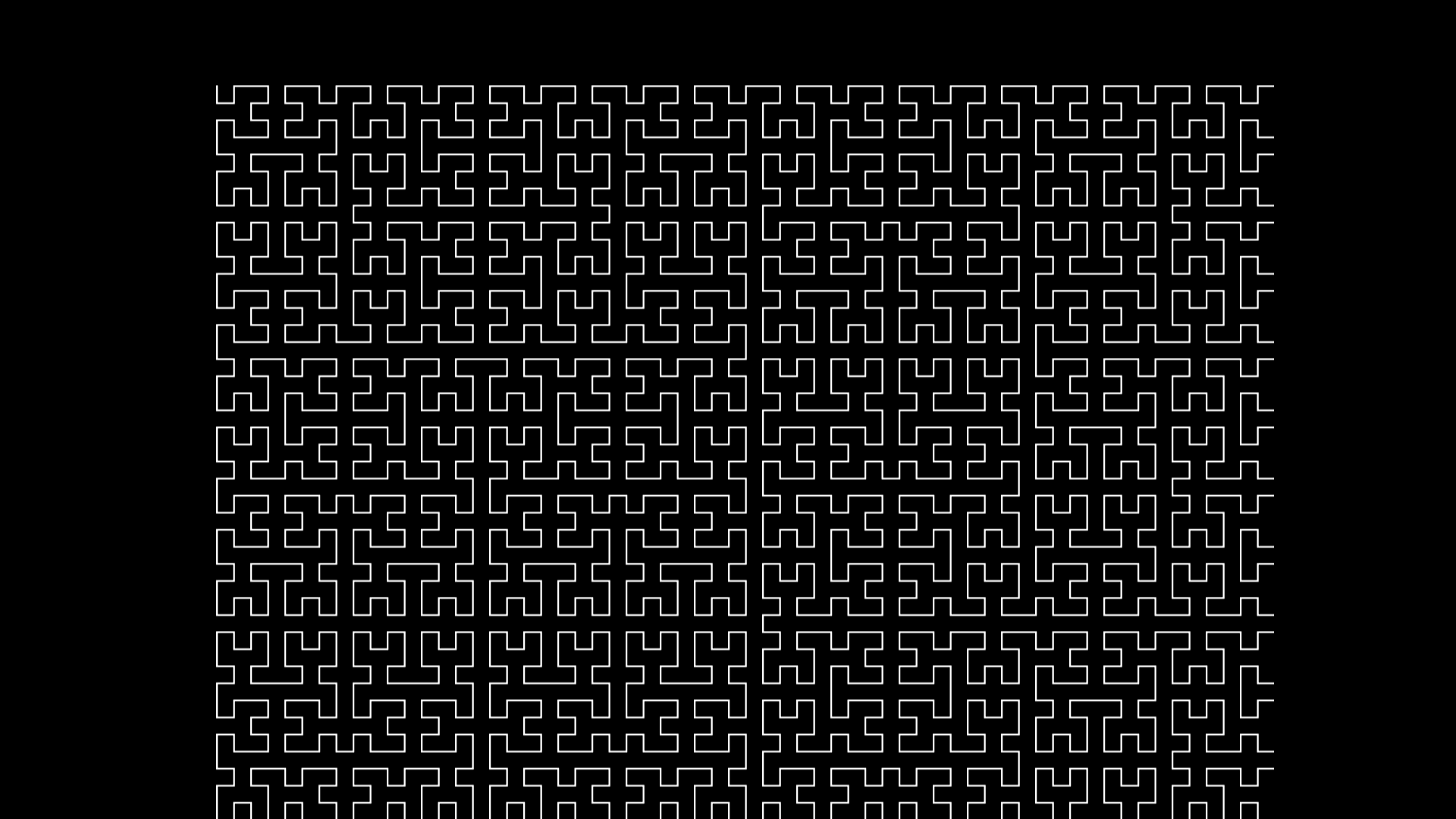
hilbert(r,d,l,u,n,h,x,y);

getch();

closegraph();

return 0;

}

Output:-            

**PROGRAM 6**

Name:- Kaustubh S Kabra

Class:-Second Year Engineering Comp-1

Roll No:- 20

**Write C++ program to simulate any one of or similar scene**

**a) Clock with pendulum OR**

**b) National Flag hoisting OR**

**c) Vehicle/boat locomotion OR**

**d) Water drop falling into the water and generated waves after impact**

Program:-

#ifdef \_\_APPLE\_\_

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#endif

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

int i, j = 0;

char st1[] = "Vande Mataram";

float w = 0.0, h = 0.0, x, y, z;

void filcircle(float x, float y, float r)

{

float angle = 0;

glBegin(GL\_TRIANGLE\_FAN);

while (angle < 360)

{

glVertex2f(x + sin(angle) \* r, y + cos(angle) \* r);

angle += 1.0;

}

glEnd();

}

void curves2(float x1, float y1, float x2, float y2, float a1, float a2, float b1, float b2)

{

GLfloat cp[4][3] = { {x1,y1},{a1,a2},{b1,b2},{x2,y2} };

glMap1f(GL\_MAP1\_VERTEX\_3, 0.0, 1.0, 3, 4, \*cp);

glEnable(GL\_MAP1\_VERTEX\_3);

GLint k;

float c = 0.3;

glLineWidth(2);

glBegin(GL\_LINE\_STRIP);

for (k = 0; k <= 50; k++)

{

glEvalCoord1f(GLfloat(k) / 50.0);

}

glEnd();

glColor3f(0.15, .3, 0.65);

glBegin(GL\_POINTS);

for (k = 0; k < 4; k++)

glVertex2fv(&cp[k][0]);

glEnd();

}

void drawline(float x0, float y0, float x1, float y1)

{

glBegin(GL\_LINES);

glVertex2f(x0, y0);

glVertex2f(x1, y1);

glEnd();

glFlush();

}

void drawrect(float xmin, float xmax, float ymin, float ymax)

{

glBegin(GL\_QUADS);

glVertex2f(xmin, ymin);

glVertex2f(xmin, ymax);

glVertex2f(xmax, ymax);

glVertex2f(xmax, ymin);

glEnd();

glFlush();

}

void draw\_pixels(int cx, int cy)

{

glPointSize(5.0);

glBegin(GL\_POINTS);

glVertex2i(cx, cy);

glEnd();

glFlush();

}

void plotpixels(int h, int k, int x, int y)

{

draw\_pixels(x + h, y + k);

draw\_pixels(-x + h, y + k);

draw\_pixels(x + h, -y + k);

draw\_pixels(-x + h, -y + k);

draw\_pixels(y + h, x + k);

draw\_pixels(-y + h, x + k);

draw\_pixels(y + h, -x + k);

draw\_pixels(-y + h, -x + k);

}

void drawcircle(int h, int k, int r)

{

int d = 1 - r, x = 0, y = r;

while (y > x)

{

plotpixels(h, k, x, y);

if (d < 0)

d += 2 \* x + 3;

else

{

d += 2 \* (x - y) + 5;

--y;

}

++x;

}

plotpixels(h, k, x, y);

}

void init()

{

glClearColor(0.15, .3, .65, 1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

void sun()

{

//drawing a sun

glColor3f(1, 1, 0);

glLineWidth(2);

glBegin(GL\_LINES);

glVertex2f(60, 63);

glVertex2f(100, 63);

glVertex2f(80, 50);

glVertex2f(80, 80);

glVertex2f(70, 77);

glVertex2f(90, 50);

glVertex2f(70, 50);

glVertex2f(90, 77);

glVertex2f(65, 70);

glVertex2f(95, 55);

glVertex2f(65, 55);

glVertex2f(95, 70);

glEnd();

float angle = 0;

glColor3f(1, .7, 0);

glBegin(GL\_TRIANGLE\_FAN);

while (angle < 360)

{

glVertex2f(80 + sin(angle) \* 10, 65 + cos(angle) \* 10);

angle += 1.0;

}

glEnd();

}

void draw\_flag()

{

//steps

glColor3f(1.5, 1.5, 1.5);

drawrect(-100, -60, -75, -70);

drawrect(-95, -65, -70, -65);

//pole

glColor3f(0.0, 0.0, 0.0);

glLineWidth(8.0);

drawline(-80, -65, -80, 55);

//rope

glColor3f(1.0, 1.0, 1.0);

glLineWidth(0.2);

curves2(-80, -40, -80, 55, -83, -30, -83, 40);

//folded flag

glColor3f(1.0, 1.0, 0.0);

drawrect(-80, -75, -42, -38);

//tierope

glColor3f(1.0, 1.0, 1.0);

glLineWidth(0.2);

drawline(-80, -40, -75, -40);

//hook

glColor3f(0.0, 0.0, 0.0);

draw\_pixels(-80, 55);

}

void draw\_people()

{

int j = 0;

float k = 0;

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

{

//person head

glColor3f(0.75, 0.75, 1);

filcircle(40 + j, -30, 6);

//neck

drawrect(38 + j, 42 + j, -40, -35);

//eye

glColor3f(0, 0, 0);

draw\_pixels(38 + j, -28);

draw\_pixels(42 + j, -28);

glLineWidth(2.0);

drawline(38 + j, -32.5, 42 + j, -32.5);

//body

glColor3f(1 + k, 0.1 + k, 0.2 + k);

drawrect(33 + j, 47 + j, -60, -40);

//hands

glColor3f(0 + k, 0.75 + k, 1 + k);

drawrect(30 + j, 33 + j, -50, -40);

drawrect(47 + j, 50 + j, -50, -40);

glColor3f(0.75, 0.75, 1);

drawrect(30 + j, 33 + j, -65, -50);

drawrect(47 + j, 50 + j, -65, -50);

//legs

glColor3f(0 + k, .75 + k, 1 + k);

drawrect(35 + j, 40 + j, -75, -60);

drawrect(40 + j, 45 + j, -75, -60);

glColor3f(0, 0, 0);

drawline(40 + j, -75, 40 + j, -60);

j = j + 25;

k += .3;

}

}

void printc(int x, int y, char st[])

{

char\* p = st;

float i = 0;

while (\*p != '\0')

{

glRasterPos2i(x + i, y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, \*p);

i += 5;

p++;

}

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

gluOrtho2D(-100.0, 100.0, -80.0 \* (GLfloat)h / (GLfloat)w,

80.0 \* (GLfloat)h / (GLfloat)w);

else

gluOrtho2D(-100.0 \* (GLfloat)w / (GLfloat)h,

100.0 \* (GLfloat)w / (GLfloat)h, -80.0, 80.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glutPostRedisplay();

}

void salute()

{

int j = 0, i;

float k = .3;

for (i = 0; i < 2; i++)

{

glColor3f(.15, 0.3, 0.65);

drawrect(55 + j, 58 + j, -50, -40);

drawrect(55 + j, 58 + j, -65, -50);

//person second

glColor3f(0 + k, 0.75 + k, 1 + k);

glBegin(GL\_QUADS);

glVertex2f(51 + j, -40);

glVertex2f(58 + j, -40);

glVertex2f(58 + j, -43);

glVertex2f(51 + j, -43);

glEnd();

glFlush();

glColor3f(0.75, 0.75, 1);

glBegin(GL\_QUADS);

glVertex2f(51 + j, -40);

glVertex2f(54 + j, -40);

glVertex2f(61 + j, -34);

glVertex2f(59 + j, -32);

glEnd();

glFlush();

j = j + 25;

k += .3;

}

glColor3f(1, 1, 1);

printc(17, 40, st1);

}

void flaghoist()

{

//flag

glColor3f(1, 0.25, 0);

drawrect(-80, -40, 49, 56);

glColor3f(1, 1, 1);

drawrect(-80, -40, 42, 49);

glColor3f(0, 1, 0);

drawrect(-80, -40, 35, 42);

glColor3f(0, 0, 1);

drawcircle(-60, 45.5, 3.5);

drawline(-60, 45.5, -56.5, 45.5);

drawline(-60, 45.5, -60, 48.5);

drawline(-60, 45.5, -60, 41.5);

drawline(-60, 45.5, -63.5, 45.5);

drawline(-60, 45.5, -58, 47);

drawline(-60, 45.5, -62, 47);

drawline(-60, 45.5, -58, 42.5);

drawline(-60, 45.5, -62, 42.5);

salute();

}

void move\_flag()

{

//hand movement

glColor3f(0.15, 0.3, 0.65);

drawrect(-70, -67, -65, -50);

glColor3f(0.75, 0.75, 1);

glBegin(GL\_QUADS);

glVertex2f(-77, -40);

glVertex2f(-78, -42);

glVertex2f(-70, -47);

glVertex2f(-70, -50);

glEnd();

glFlush();

//flag movement

float i = 0, j = 0;

while (i < 80.5)

{

glColor3f(0.15, 0.3, 0.65);

drawrect(-80, -75, -42 + i, -38 + i);

i = i + .15;

j = i;

glColor3f(1.0, 1.0, 0.0);

drawrect(-80, -75, -42 + j, -38 + j);

}

glColor3f(.15, 0.3, 0.65);

curves2(-80, -40, -80, 55, -83, -30, -83, 40);

//pole

glColor3f(0.0, 0.0, 0.0);

glLineWidth(8.0);

drawline(-80, -65, -80, 55);

glColor3f(1.0, 1.0, 1.0);

glLineWidth(0.2);

curves2(-75, -50, -80, 55, -65, -35, -95, 40);

//untie rope

glBegin(GL\_LINES);

glVertex2f(-78.50, -50);

glVertex2f(-74, -43);

glEnd();

glFlush();

glBegin(GL\_LINES);

glVertex2f(-80, -40);

glVertex2f(-74, -43);

glEnd();

glFlush();

flaghoist();

}

void move\_person()

{

int i = 0;

while (i < 100)

{

//1st person head

glColor3f(0.15, .3, .65);

filcircle(40 - i, -30, 6);

//neck

drawrect(38 - i, 42 - i, -40, -35);

//eye

draw\_pixels(38 - i, -28);

draw\_pixels(42 - i, -28);

drawline(38 - i, -32.5, 42 - i, -32.5);

//body

drawrect(33 - i, 47 - i, -60, -40);

//hands

drawrect(30 - i, 33 - i, -50, -40);

drawrect(47 - i, 50 - i, -50, -40);

drawrect(30 - i, 33 - i, -65, -50);

drawrect(47 - i, 50 - i, -65, -50);

//legs

drawrect(35 - i, 40 - i, -75, -60);

drawrect(40 - i, 45 - i, -75, -60);

drawline(40 - i, -75, 40 - i, -60);

i = i + 5;

j = i;

//1st person head

glColor3f(0.75, 0.75, 1);

filcircle(40 - j, -30, 6);

//neck

drawrect(38 - j, 42 - j, -40, -35);

//eye

glColor3f(0, 0, 0);

draw\_pixels(38 - j, -28);

draw\_pixels(42 - j, -28);

glLineWidth(2.0);

drawline(38 - j, -32.5, 42 - j, -32.5);

//body

glColor3f(1, 0.1, 0.2);

drawrect(33 - j, 47 - j, -60, -40);

//hands

glColor3f(0, 0.75, 1);

drawrect(30 - j, 33 - j, -50, -40);

drawrect(47 - j, 50 - j, -50, -40);

glColor3f(0.75, 0.75, 1);

drawrect(30 - j, 33 - j, -65, -50);

drawrect(47 - j, 50 - j, -65, -50);

//legs

glColor3f(0, .75, 1);

drawrect(35 - j, 40 - j, -75, -60);

drawrect(40 - j, 45 - j, -75, -60);

glColor3f(0, 0, 0);

drawline(40 - j, -75, 40 - j, -60);

}

move\_flag();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

sun();

draw\_flag();

draw\_people();

}

void keys(unsigned char key, int x, int y)

{

if (key == 's' || key == 'S')

move\_person();

if (key == 27)

exit(0);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_SINGLE);

glutInitWindowSize(1000, 700);

glutInitWindowPosition(0, 0);

glutCreateWindow("Flag Hoisting Ceremony");

glutDisplayFunc(display);

glutReshapeFunc(reshape);

init();

glutKeyboardFunc(keys);

glutMainLoop();

}

Output:- 