****

**RAMNIRANJAN JHUNJHUNWALA COLLEGE GHATKOPAR (W), MUMBAI - 400 086**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**2022 - 2023**

**S.Y. B. Sc.( I.T.) SEM IV**

**Paper SUIT405-COMPUTER GRAPHICS AND ANIMATION**

**Name :- Rajbhar Sudesh Dinesh SushilaDevi**

**Roll No. 3067**

**Hindi Vidya Prachar Samiti’s**

**RAMNIRANJAN JHUNJHUNWALA COLLEGE**

**Ghatkopar (W), Mumbai-400 086**

***Certificate***



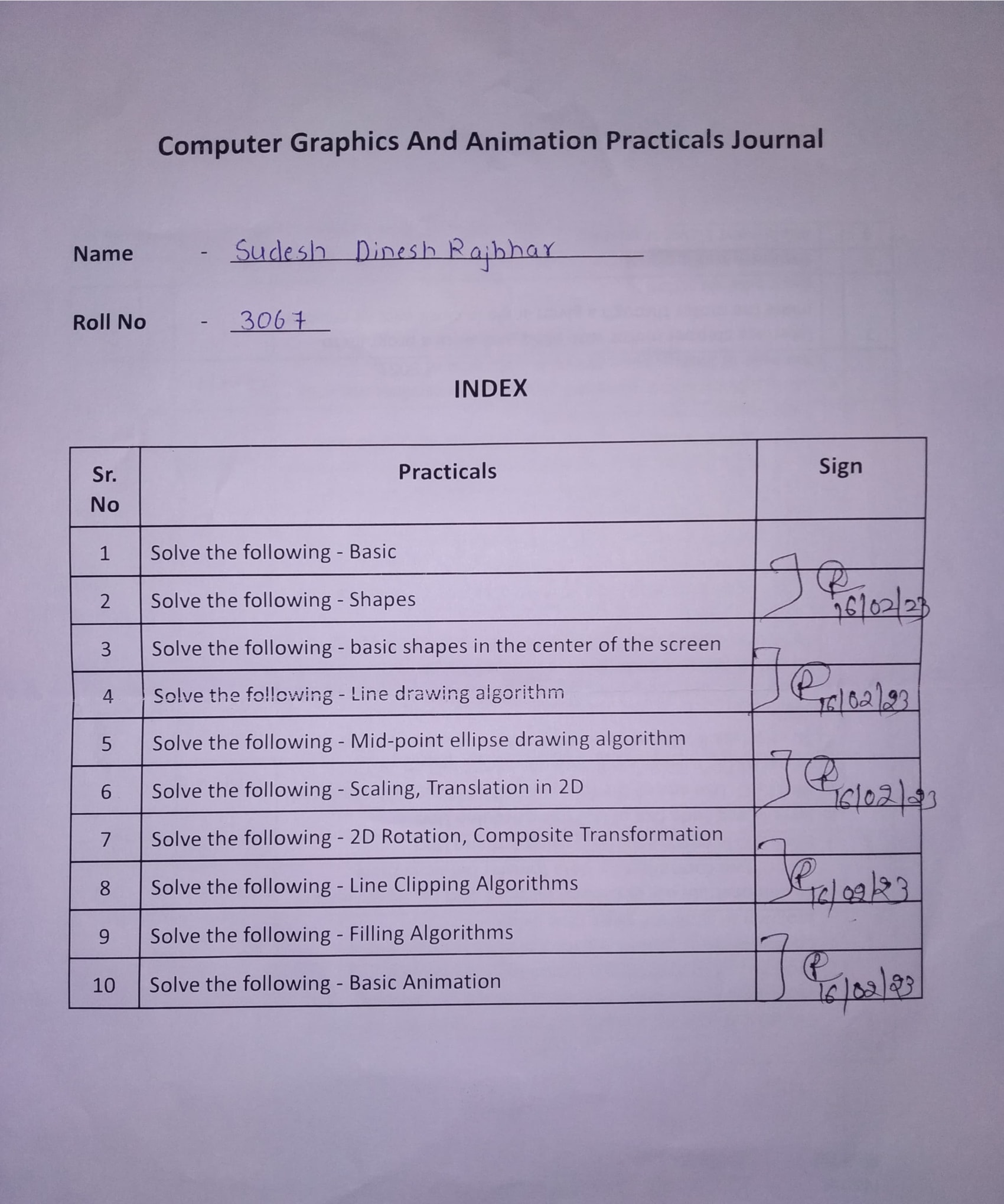
**This is to certify that Mr./Ms. Rajbhar Sudesh Dinesh SushilaDevi Roll No 3067 of S.Y.B.Sc.(I.T.) class has completed the required number of experiments in the subject of Computer Graphics and Animation in the Department of Information Technology during the academic year 2022 - 2023 .**

**Professor In-Charge Co-ordinator of IT Department**

**Prof. Bharati Bhole**

**Prof. Archana Bhide**

**College Seal & Date Examiner**

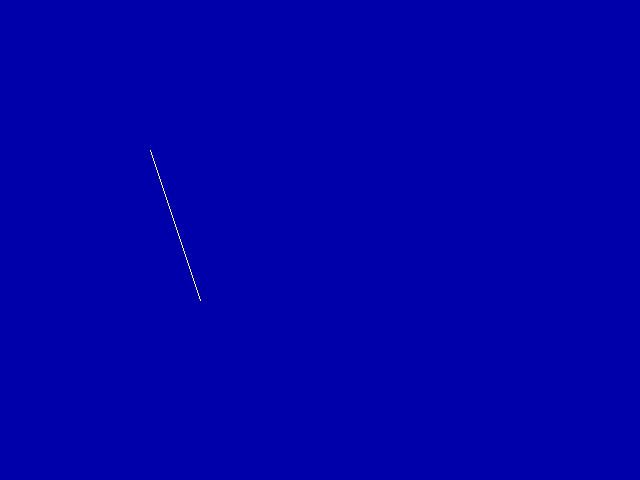
IPractical no 1:

# **1A)Study and enlist the basic functions used for graphics in C / C++ / Python language. Give an example for each of them.**

Program for LINE:

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  int main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,”C:\\TURBOC3\\BGI”);  setbkcolor(BLUE);  line(150,150,200,300);  getch();  closegraph();  } |
| --- |

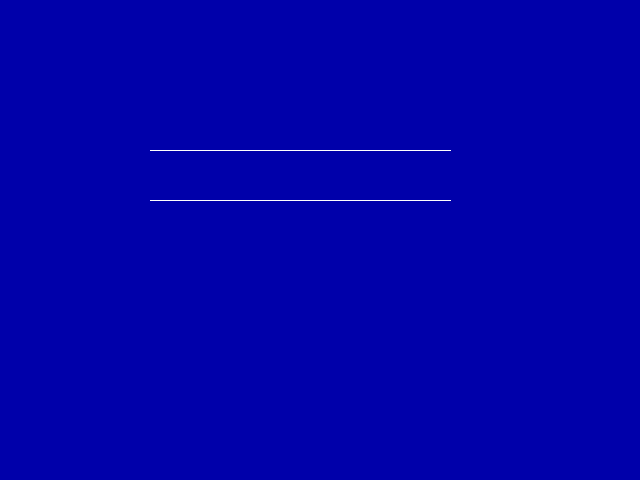
Output:



**Program for parallel LINE:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  line(150,150,450,150);  line(150,200,450,200);  getch();  closegraph();  } |
| --- |

Output:



**Program for putpixel:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  //setcolor(RED);  putpixel(200,100,RED);  getch();  closegraph();  } |
| --- |

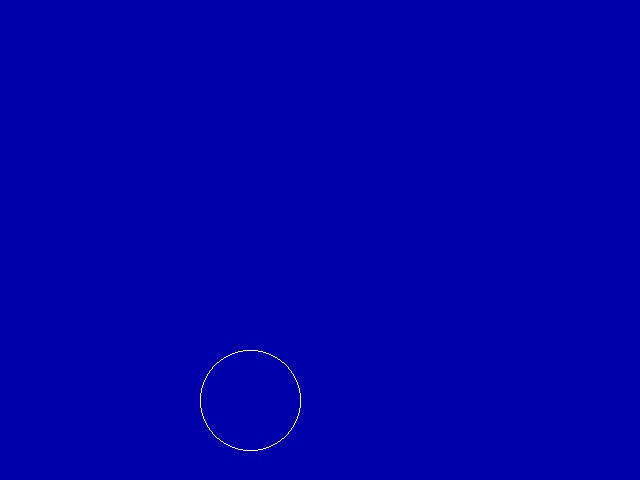
Output:



**Program for Circle:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  circle(250,400,50);  getch();  closegraph();  } |
| --- |

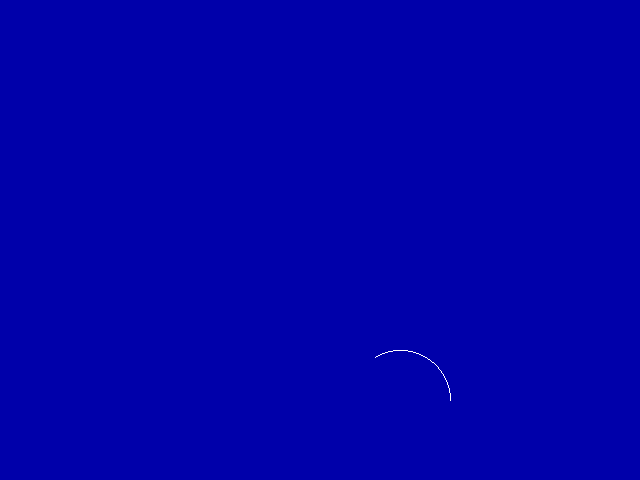
Output:



**Program for Arc:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  arc(400,400,0,120,50);  getch();  closegraph();  } |
| --- |

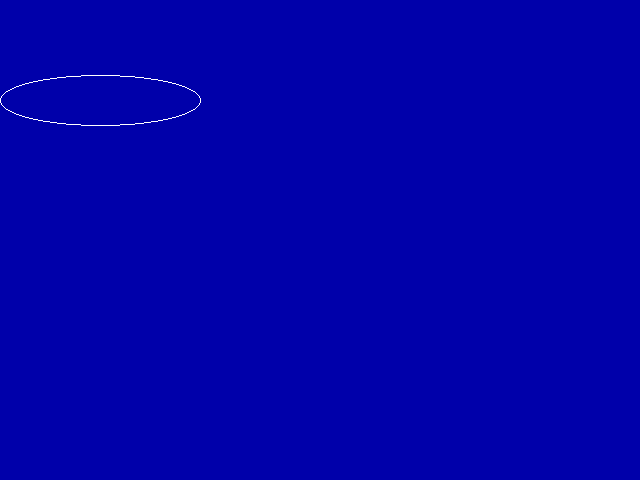
Output:



**Program for Ellipse:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  ellipse(100,100,0,360,100,25);  getch();  closegraph();  } |
| --- |

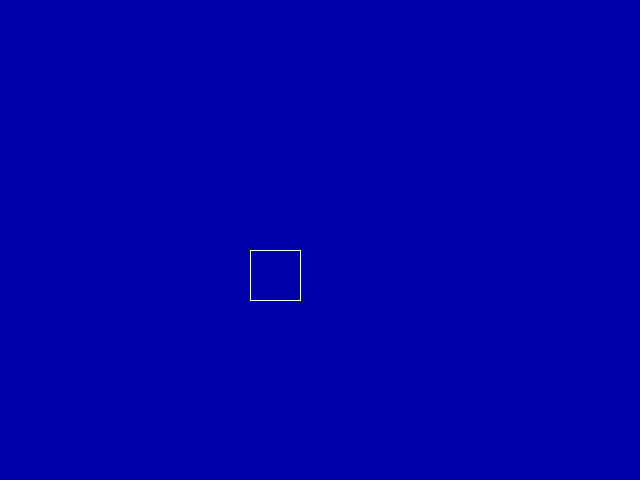
Output:



**Program for Rectangle:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  rectangle(300,300,250,250);  getch();  closegraph();  } |
| --- |

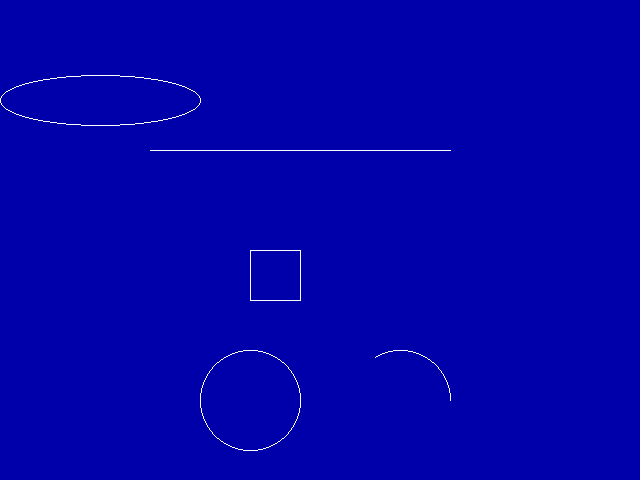
Output:



**Program with multiple methods:**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(BLUE);  putpixel(200,100,RED);  line(150,150,450,150);  line(150,150,450,150);  circle(250,400,50);  rectangle(300,300,250,250);  arc(400,400,0,120,50);  ellipse(100,100,0,360,100,25);  getch();  closegraph();  } |
| --- |

Output:



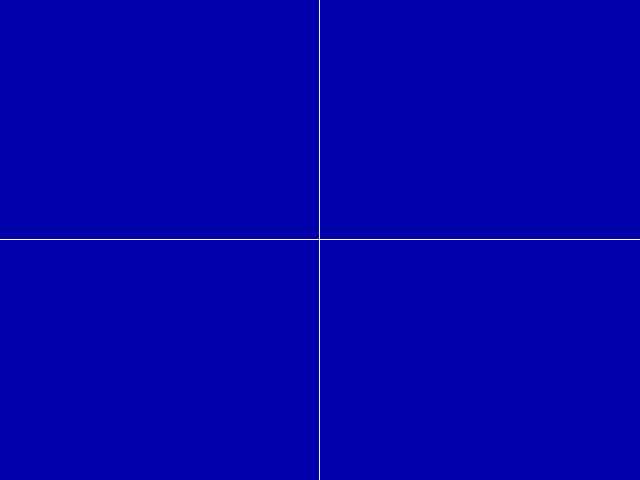
# 

# **1B) Draw COORDINATE System through the center of the screen.**

Program:

| #include<stdio.h>  #include<stdlib.h>  #include<conio.h>  #include<graphics.h>  int main()  {  int gd = DETECT,gm;  int xmax,ymax;  initgraph(&gd,&gm,"C:\\TurboC3\\BGI");  setbkcolor(BLUE);  xmax = getmaxx();  ymax = getmaxy();  line(xmax/2,0,xmax/2,ymax);  line(0,ymax/2,xmax,ymax/2);  getch();  closegraph();  return 0;  } |
| --- |

Output:



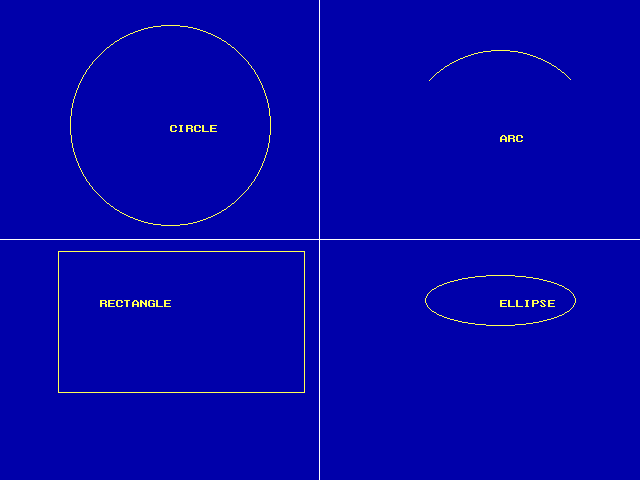
**Practical No 2 :**

# 2A)Divide your screen into four regions, draw a circle, rectangle, ellipse and half ellipse in each region with appropriate messages.

**Source Code -**

| #include<graphics.h>  #include<conio.h>  void main()  {int gd=DETECT,gm;  int xmax,ymax;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  xmax=getmaxx();  ymax=getmaxy();  line(xmax/2,0,xmax/2,ymax);  line(0,ymax/2,xmax,ymax/2);  setbkcolor(BLUE);  setcolor(YELLOW);  circle(170,125,100);  outtextxy(170,125,"CIRCLE");  setcolor(YELLOW);  rectangle(58,251,304,392);  outtextxy(100,300,"RECTANGLE");  setcolor(YELLOW);  arc(500,150,45,135,100);  outtextxy(500,135,"ARC");  setcolor(YELLOW);  ellipse(500,300,0,360,75,25);  outtextxy(500,300,"ELLIPSE");  getch();  } |
| --- |

Output:



# 

# 2B)Draw a simple hut on the screen.

Program:

| #include<graphics.h>  #include<conio.h>  #include<stdio.h>  void main()  {  int gd=DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*HUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t");  setcolor(YELLOW);  rectangle(174,201,276,393);  outtextxy(180,203,"SUDESH HUT");  rectangle(201,299,246,393);  rectangle(276,201,440,393);  line(174,201,224,120);  line(274,201,224,120);  line(224,120,405,120);  line(405,120,440,201);  circle(230,340,7);  circle(351,279,18);  circle(351,279,43);  getch();  closegraph();  } |
| --- |

Output:



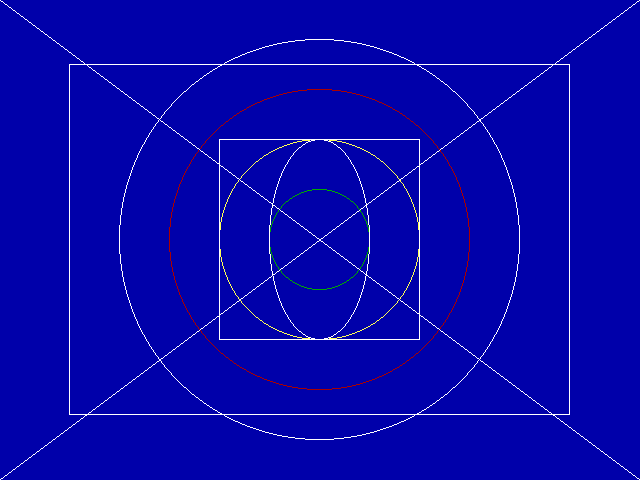
Practical 3 :

# 3A )Draw the following basic shapes in the center of the screen:

Code:

| #include<graphics.h>  #include<stdio.h>  #include<conio.h>  void main(){  int gd = DETECT,gm;  int midx,midy;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  midx = getmaxx()/2;  midy = getmaxy()/2;  setbkcolor(BLUE);  setcolor(WHITE);  circle(midx,midy,200);  setcolor(RED);  circle(midx,midy,150);  setcolor(YELLOW);  circle(midx,midy,100);  setcolor(GREEN);  circle(midx,midy,50);  setcolor(WHITE);  rectangle(midx-100,midy-100,midx+100,midy+100);  rectangle(midx-250,midy-175,midx+250,midy+175);  ellipse(midx,midy,0,360,50,100);  line(0,0,getmaxx(),getmaxy());  line(0,getmaxy(),getmaxx(),0);  getch();  closegraph();  return ;  } |
| --- |

Output:



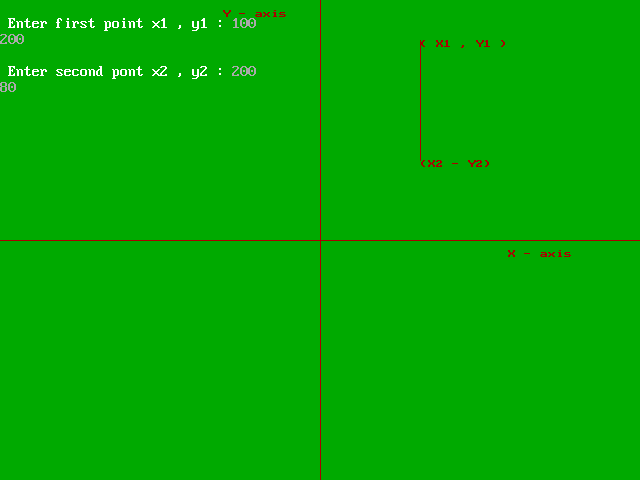
Practical 4 :

# 4A)Develop the program for the DDA Line drawing algorithm.

Code:

| #include<graphics.h>  #include<conio.h>  #include<iostream.h>  #include<dos.h>  #include<math.h>  void ddaline(int x1,int y1,int x2,int y2);  void main()  {  int x1,x2,y1,y2;  int gd = DETECT,gm,xmid,ymid;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  setbkcolor(GREEN);  setcolor(RED);  cout<<"\n Enter x1 , y1 : ";  cin>>x1>>y1;  cout<<"\n Enter x2 , y2 : ";  cin>>x2>>y2;  line(0,240,640,240);  line(320,0,320,480);  outtextxy(590,250," X - axis ");  outtextxy(300,0," Y - axis ");  ddaline(x1,x2,y1,y2);  getch();  cosegraph();  }  void ddaline(int x1,int y1,int x2,int y2)  {  int dx,dy,step,k;  float xinc,yinc,x,y;  dx=x2-x1;  dy=y2-y1;  x=x1;  y=y1;  if(abs(dx)>=abs(dy))  { step = abs(dx);}  else  { step=abs(dy);}  xinc = dx/step;  yinc = dy/step;  putpixel(ceil(x)+320,240-ceil(y),RED);  outtextxy(320+x,240-y,"( X1 , Y1 )");  for(k=1;k<=step;k++)  {  x=x+xinc;  y=y+yinc;  delay(100);  putpixel(320+ceil(x),240-ceil(y),RED);  }  outtextxy(320+x,240-y,"(X2 - Y2)");  } |
| --- |

output:

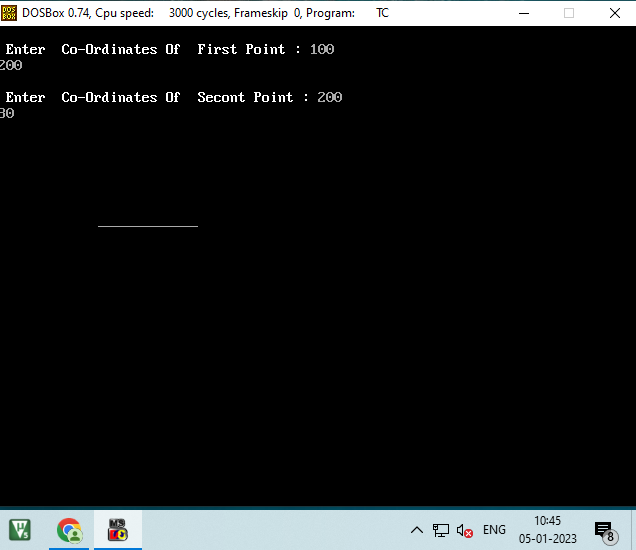


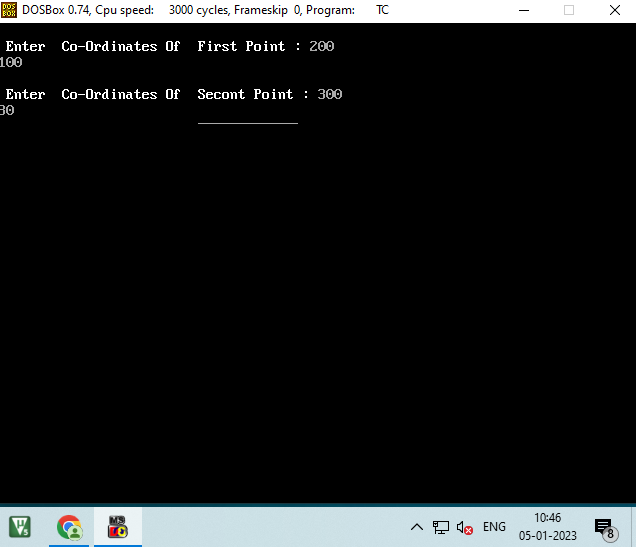
# 4B) Develop the program for Bresenham’s Line drawing algorithm

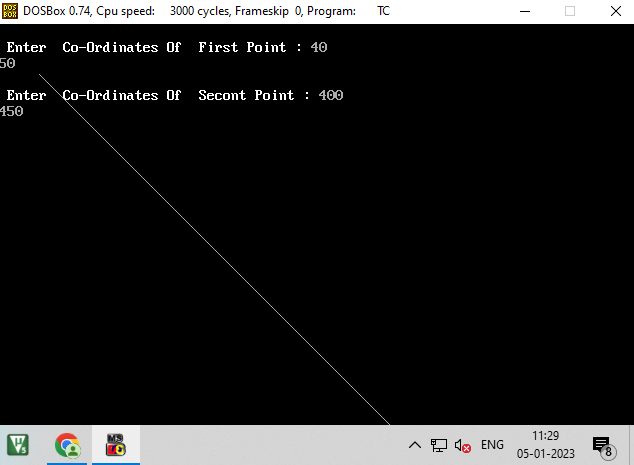
**Source Code -**

| #include<iostream.h>  #include<graphics.h>  #include<conio.h>  void drawline(int x0,int y0,int x1,int y1)  { int dx,dy,p,x,y;  dx=x1-x0;  dy=y1-y0;  x=x0;  y=y0;  p=2\*dy-dx;  while(x<x1)  {  if(p>=0)  {  putpixel(x,y,7);  y=y+1;  p=p+2\*dy-2\*dx;  }  else{  putpixel(x,y,7);  p=p+2\*dy;  }  x=x+1;  }  }  void main()  {  int x0,y0,x1,y1;  int gd = DETECT,gm,error;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  // setbkcolor(WHITE);  setcolor(CYAN);  cout<<"\n Enter Co-Ordinates Of First Point : ";  cin>>x0>>y0;  cout<<"\n Enter Co-Ordinates Of Secont Point : ";  cin>>x1>>y1;  drawline(x0,y0,x1,y1);  getch();  closegraph();  } |
| --- |

Output:







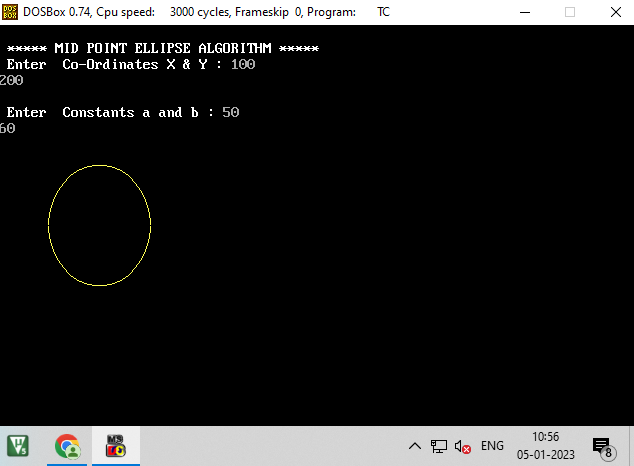
Practical 5 :

# 5A)Develop the program for the mid-point ellipse drawing algorithm.

**Source Code -**

| #include<stdio.h>  #include<dos.h>  #include<iostream.h>  #include<graphics.h>  #include<conio.h>  void main()  {  long x,y,x\_center,y\_center;  long a\_sqr,b\_sqr,fx,fy,d,a,b,tmp1,tmp2;  int gd = DETECT,gm;  clrscr();  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  cout<<"\n \*\*\*\*\* MID POINT ELLIPSE ALGORITHM \*\*\*\*\*";  cout<<"\n Enter Co-Ordinates X & Y : ";  cin>>x\_center>>y\_center;  cout<<"\n Enter Constants a and b : ";  cin>>a>>b;  x=0;  y=b;  a\_sqr = a\*a;  b\_sqr=b\*b;  fx=2\*b\_sqr\*x;  fy=2\*a\_sqr\*y;  d = b\_sqr - (a\_sqr \* b) + (a\_sqr \* 0.25);  do  {  putpixel(x\_center +x , y\_center +y , 14);  putpixel(x\_center -x , y\_center -y , 14);  putpixel(x\_center +x , y\_center -y , 14);  putpixel(x\_center -x , y\_center +y , 14);  if(d<0)  {  d = d + fx + b\_sqr;  }  else  {  y = y-1;  d= d+fx+-fy+b\_sqr;  fy=fy - (2\*a\_sqr);  }  x=x+1;  fx=fx+(2\*b\_sqr);  delay(10);  }  while (fx<fy);  tmp1 =(x+0.5) \* (x+0.5) ;  tmp2 = (y-1) \* (y-1) ;  d = b\_sqr \* tmp1+ a\_sqr\* tmp2 - (a\_sqr \* b\_sqr);  do  {  putpixel(x\_center + x , y\_center + y ,14);  putpixel(x\_center - x , y\_center - y ,14);  putpixel(x\_center + x , y\_center - y ,14);  putpixel(x\_center - x , y\_center + y ,14);  if(d>=0)  d= d-fy+a\_sqr;  else  {  x=x+1;  d= d+fx - fy +a\_sqr;  fx=fx+(2\*b\_sqr);  }  y=y-1;  fy =fy-(2\*a\_sqr);  }  while ( y > 0);  getch();  closegraph();  } |
| --- |

Output:



Practical 6 - Solve The following

------------------------------------------------------------------------------------------------------

# 6A.Write a program to implement 2D Translation.

SOURCE CODE:

| #include<iostream.h>  #include<graphics.h>  #include<conio.h>  void main()  {  int x,y,x1,y1,x2,y2,tx,ty,x3,y3,x4,y4;  int gd = DETECT,gm;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  clrscr();  cout<<"\n Enter 2 Lines End Points : ";  cin>>x1>>y1>>x2>>y2;  line(x1,y1,x2,y2);  cout<<"\n Enter Translation Coordinates : ";  cin>>x>>y;  x3=x1+tx;  y3=y1+ty;  x4=x2+tx;  y4=x2+ty;  cout<<"\n Line After Transaltion ";  line(x3,y3,x4,y4);  getch();  closegraph();  } |
| --- |

Output:

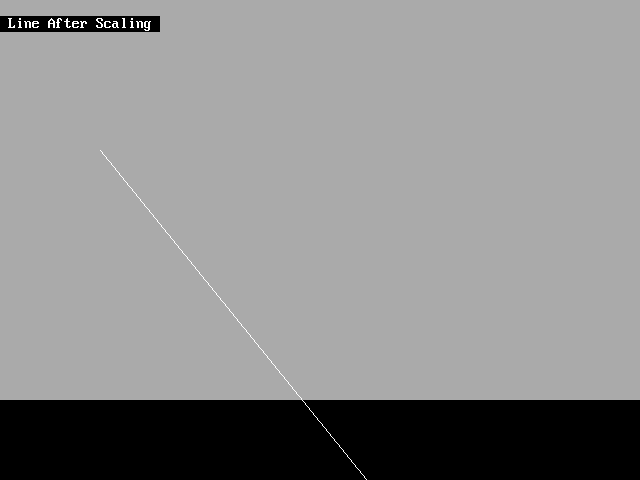


# 6B.Write a program to implement 2D Scaling.

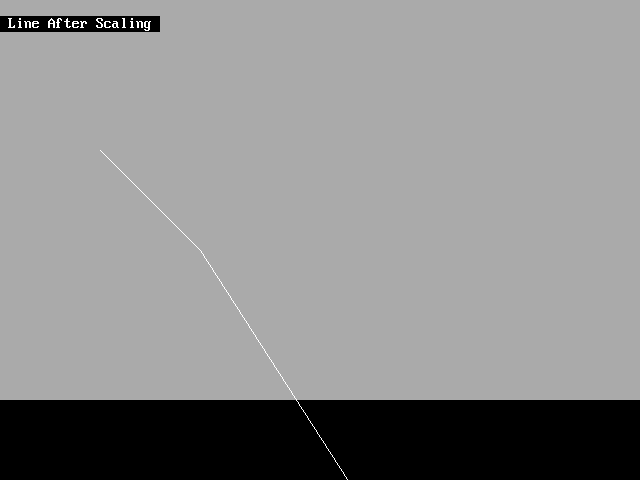
Source Code:

| #include<iostream.h>  #include<graphics.h>  #include<conio.h>  void main()  {  int x1,y1,x2,y2,x,y,x3,y3,x4,y4;  int gd = DETECT,gm;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  clrscr();  cout<<"\n Enter 2 Lines End Points : ";  cin>>x1>>y1>>x2>>y2;  line(x1,y1,x2,y2);  cout<<"\n Enter Scaling Coordinates : ";  cin>>x>>y;  x3=x1\*x;  y3=y1\*y;  x4=x2\*x;  y4=y2\*y;  cout<<"\n Line After Scaling ";  //line(x3,y3,x2,y2);  line(x1,y1,x4,y4);  getch();  closegraph();  } |
| --- |

OUTPUT1 :



Output 2:



Practical 7 - Solve The following

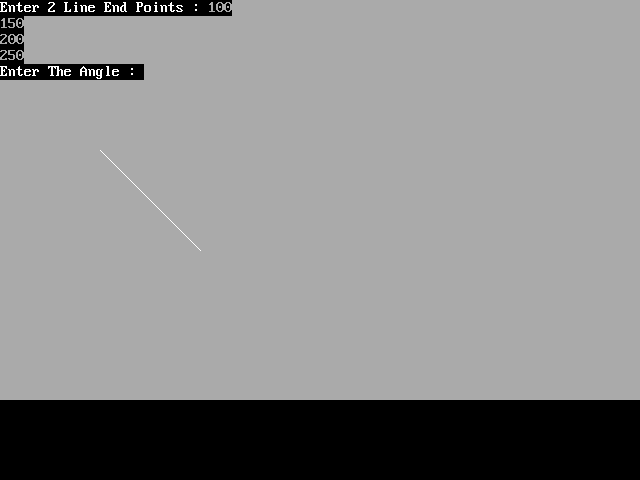
------------------------------------------------------------------------------------------------------

# A.Perform 2D Rotation on a given object.

Source code:

| #include<iostream.h>  #include<graphics.h>  #include<conio.h>  #include<stdlib.h>  #include<math.h>  void main()  {  int gd = DETECT,gm;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  int x1,y1,x2,y2,x,y,xn,yn;  double r11,r12,r21,r22,th;  clrscr();  cout<<"Enter 2 Line End Points : ";  cin>>x1>>y1>>x2>>y2;  line(x1,y1,x2,y2);  cout<<"Enter The Angle : ";  cin>>th;  r11 = cos((th\*3.1428)/180);  r12 = sin((th\*3.1428)/180);  r21 = (-sin((th\*3.1428)/180));  r22 = (-cos((th\*3.1428)/180));  xn = ((x2\*r11) - (y2 \* r12 ));  yn = ((x2 \* r21) + (y2 \* r22));  //clrscr();  cout<<"\nLine After Rotation ";  line(x1,y1,xn,yn);  getch();  closegraph();  } |
| --- |

OUTPUT1 :



OUTPUT 2 :





# B.Program to create a house like figure and perform the following operations

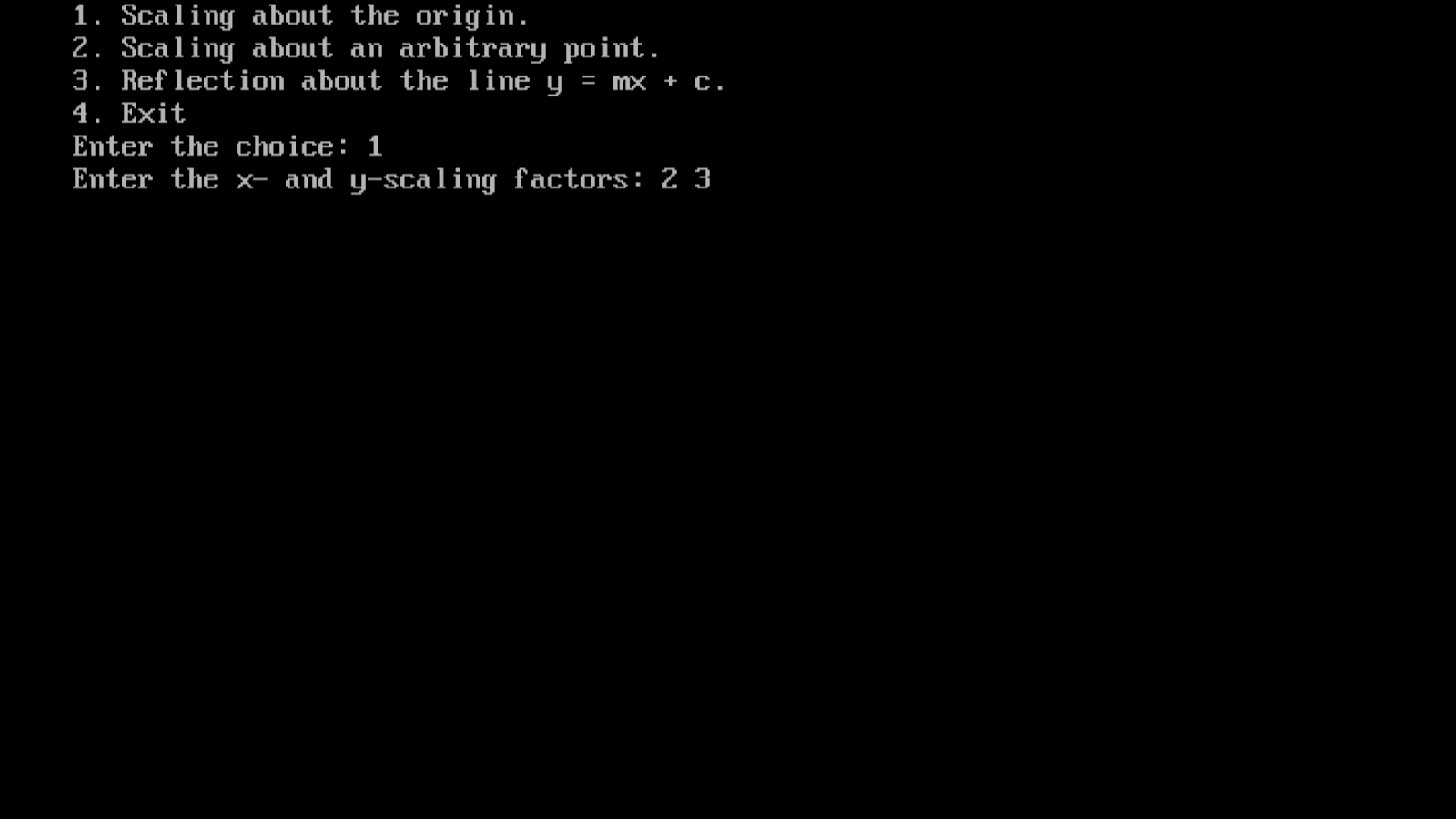
* Scaling about the origin followed by translation.
* Scaling with reference to an arbitrary point.
* Reflect about the line y = mx + c.

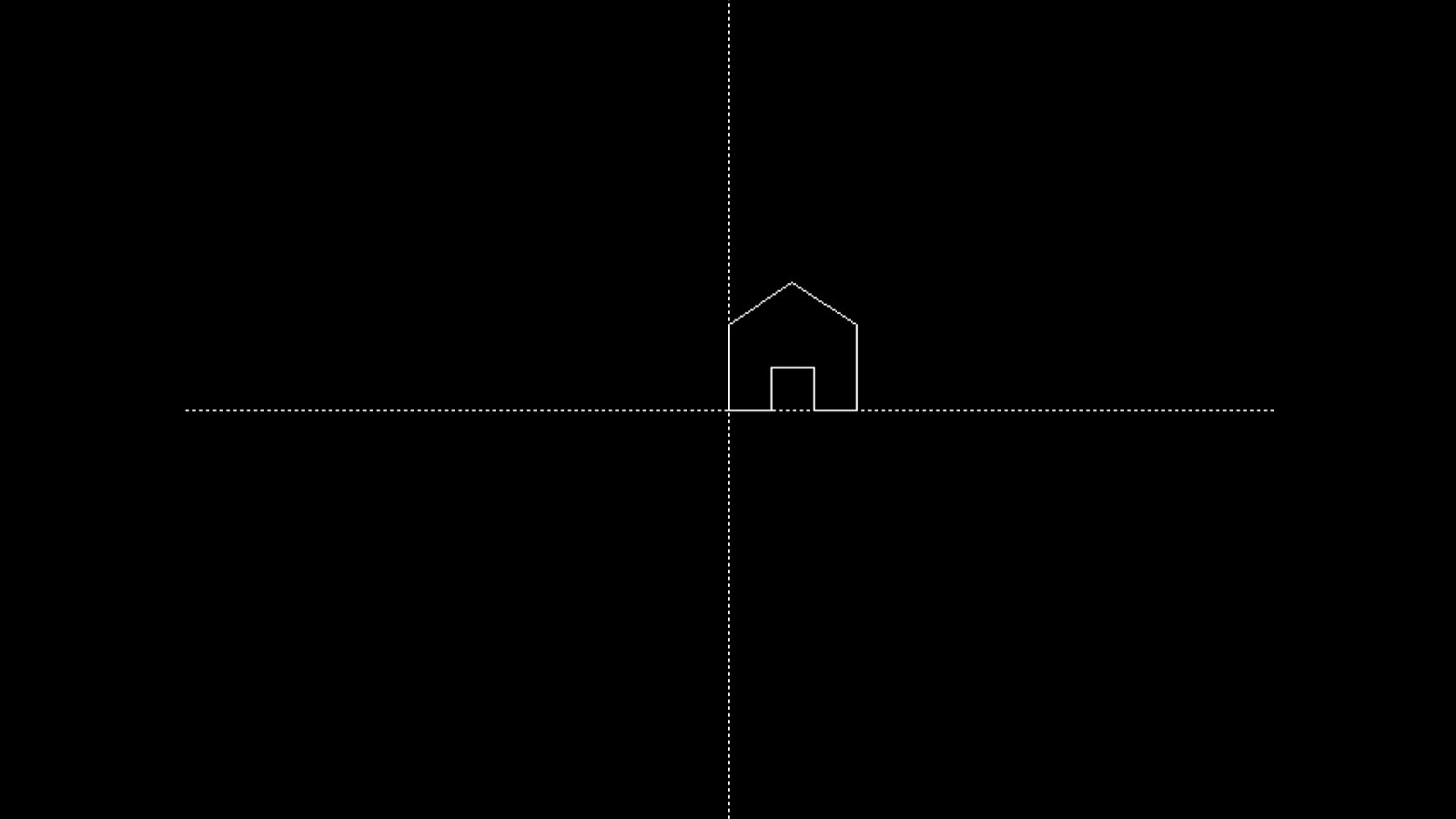
**Source Code -**

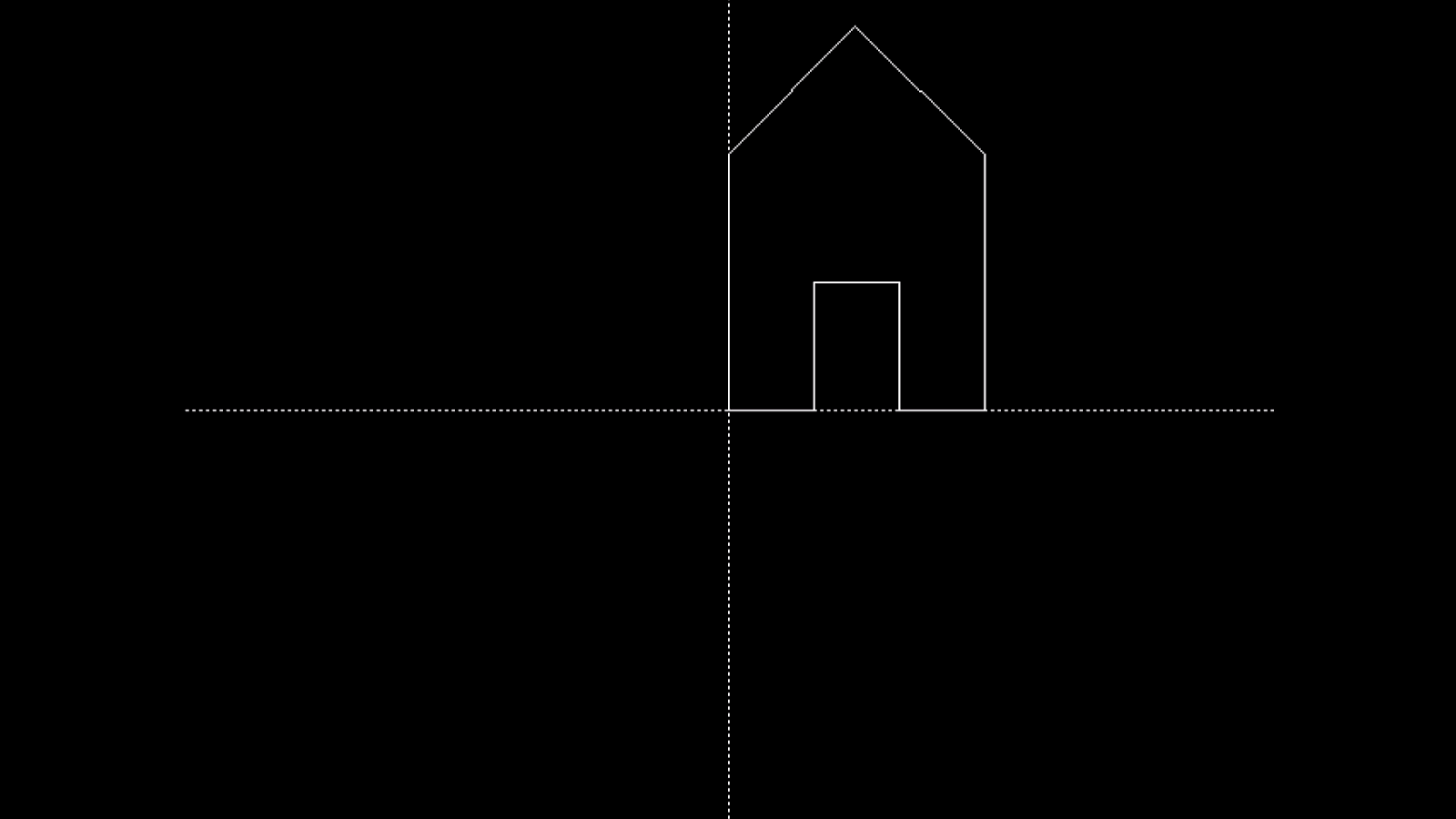
| /\* C Program to create house and perform operations in Graphics \*/  #include <stdio.h>  #include <graphics.h>  #include <stdlib.h>  #include <math.h>  #include <conio.h>    void reset (int h[][2])  {  int val[9][2] = {  { 50, 50 },{ 75, 50 },{ 75, 75 },{ 100, 75 },  { 100, 50 },{ 125, 50 },{ 125, 100 },{ 87, 125 },{ 50, 100 }  };  int i;  for (i=0; i<9; i++)  {  h[i][0] = val[i][0]-50;  h[i][1] = val[i][1]-50;  }  }  void draw (int h[][2])  {  int i;  setlinestyle (DOTTED\_LINE, 0, 1);  line (320, 0, 320, 480);  line (0, 240, 640, 240);  setlinestyle (SOLID\_LINE, 0, 1);  for (i=0; i<8; i++)  line (320+h[i][0], 240-h[i][1], 320+h[i+1][0], 240-h[i+1][1]);  line (320+h[0][0], 240-h[0][1], 320+h[8][0], 240-h[8][1]);  }  void rotate (int h[][2], float angle)  {  int i;  for (i=0; i<9; i++)  {  int xnew, ynew;  xnew = h[i][0] \* cos (angle) - h[i][1] \* sin (angle);  ynew = h[i][0] \* sin (angle) + h[i][1] \* cos (angle);  h[i][0] = xnew; h[i][1] = ynew;  }  }  void scale (int h[][2], int sx, int sy)  {  int i;  for (i=0; i<9; i++)  {  h[i][0] \*= sx;  h[i][1] \*= sy;  }  }  void translate (int h[][2], int dx, int dy)  {  int i;  for (i=0; i<9; i++)  {  h[i][0] += dx;  h[i][1] += dy;  }  }  void reflect (int h[][2], int m, int c)  {  int i;  float angle;  for (i=0; i<9; i++)  h[i][1] -= c;  angle = M\_PI/2 - atan (m);  rotate (h, angle);  for (i=0; i<9; i++)  h[i][0] = -h[i][0];  angle = -angle;  rotate (h, angle);  for (i=0; i<9; i++)  h[i][1] += c;  }    void ini()  {  int gd=DETECT,gm;  initgraph(&gd,&gm,"..\\bgi");  }  void dini()  {  getch();  closegraph();  }  void main()  {    int h[9][2],sx,sy,x,y,m,c,choice;  do  {  clrscr();  printf("1. Scaling about the origin.\n");  printf("2. Scaling about an arbitrary point.\n");  printf("3. Reflection about the line y = mx + c.\n");  printf("4. Exit\n");  printf("Enter the choice: ");  scanf("%d",&choice);  switch(choice)  {  case 1: printf ("Enter the x- and y-scaling factors: ");  scanf ("%d%d", &sx, &sy);  ini();  reset (h);  draw (h);getch();  scale (h, sx, sy);  cleardevice();  draw (h);  dini();  break;    case 2: printf ("Enter the x- and y-scaling factors: ");  scanf ("%d%d", &sx, &sy);  printf ("Enter the x- and y-coordinates of the point: ");  scanf ("%d%d", &x, &y);  ini();  reset (h);  translate (h, x, y);// Go to arbitrary point  draw(h); getch();//Show its arbitrary position  cleardevice();  translate(h,-x,-y);//Take it back to origin  draw(h);  getch();  cleardevice();  scale (h, sx, sy);//Now Scale it  draw(h);  getch();  translate (h, x, y);//Back to Arbitrary point  cleardevice();  draw (h);  putpixel (320+x, 240-y, WHITE);  dini();  break;    case 3: printf ("Enter the values of m and c: ");  scanf ("%d%d", &m, &c);  ini();  reset (h);  draw (h); getch();  reflect (h, m, c);  cleardevice();  draw (h);  dini();  break;    case 4: exit(0);  }  }while(choice!=4);  } |
| --- |

Output:

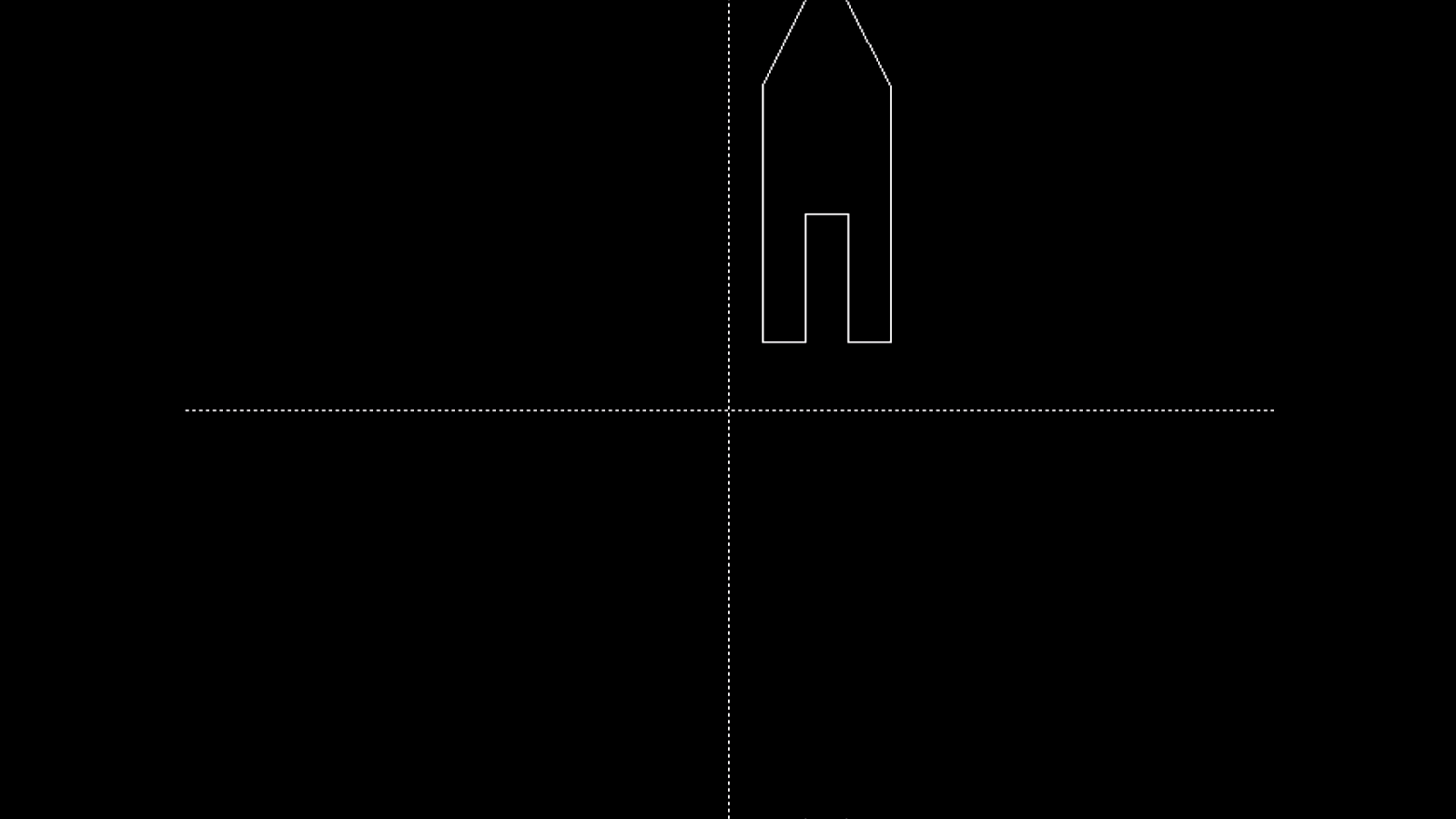
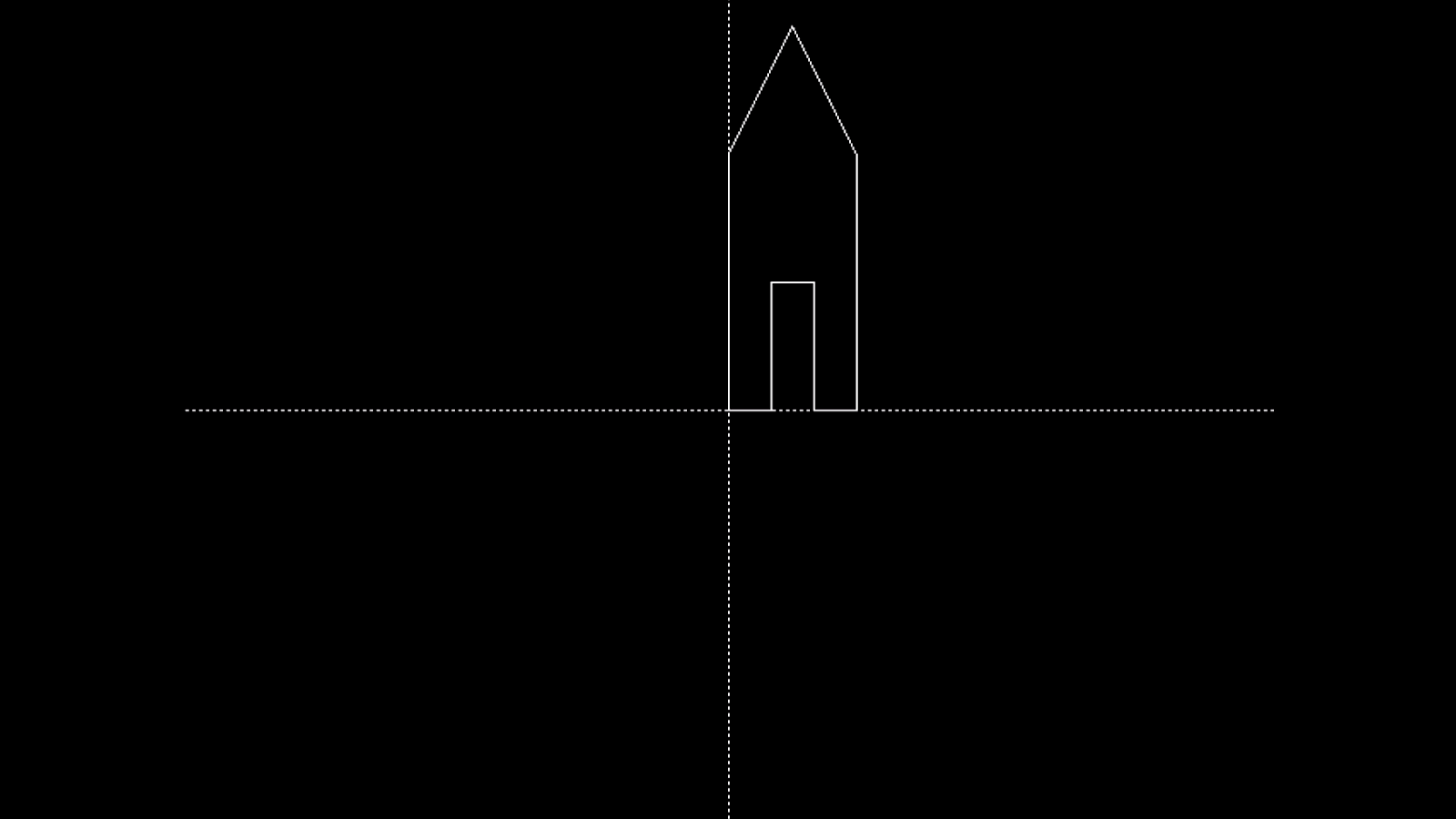
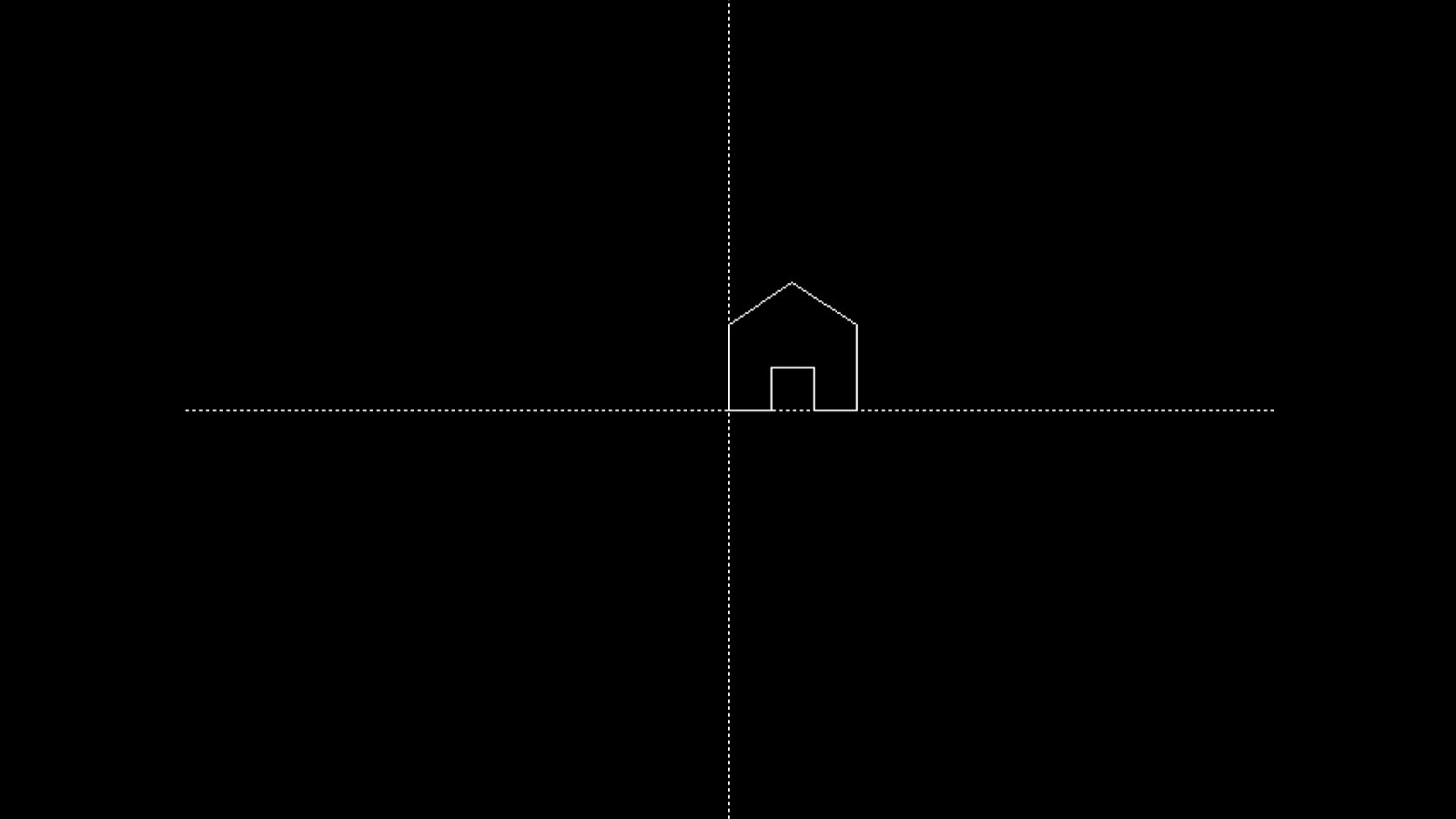
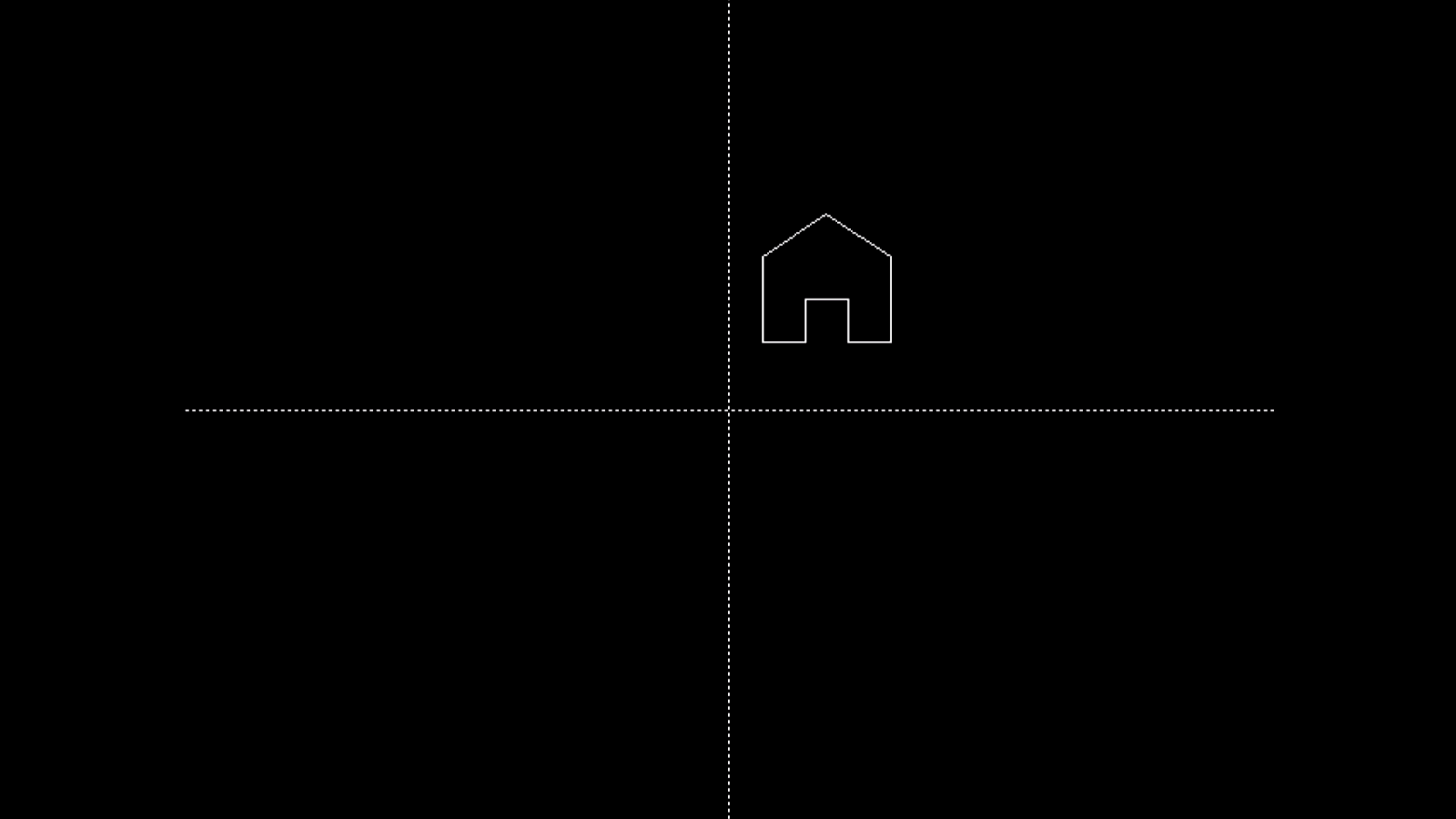
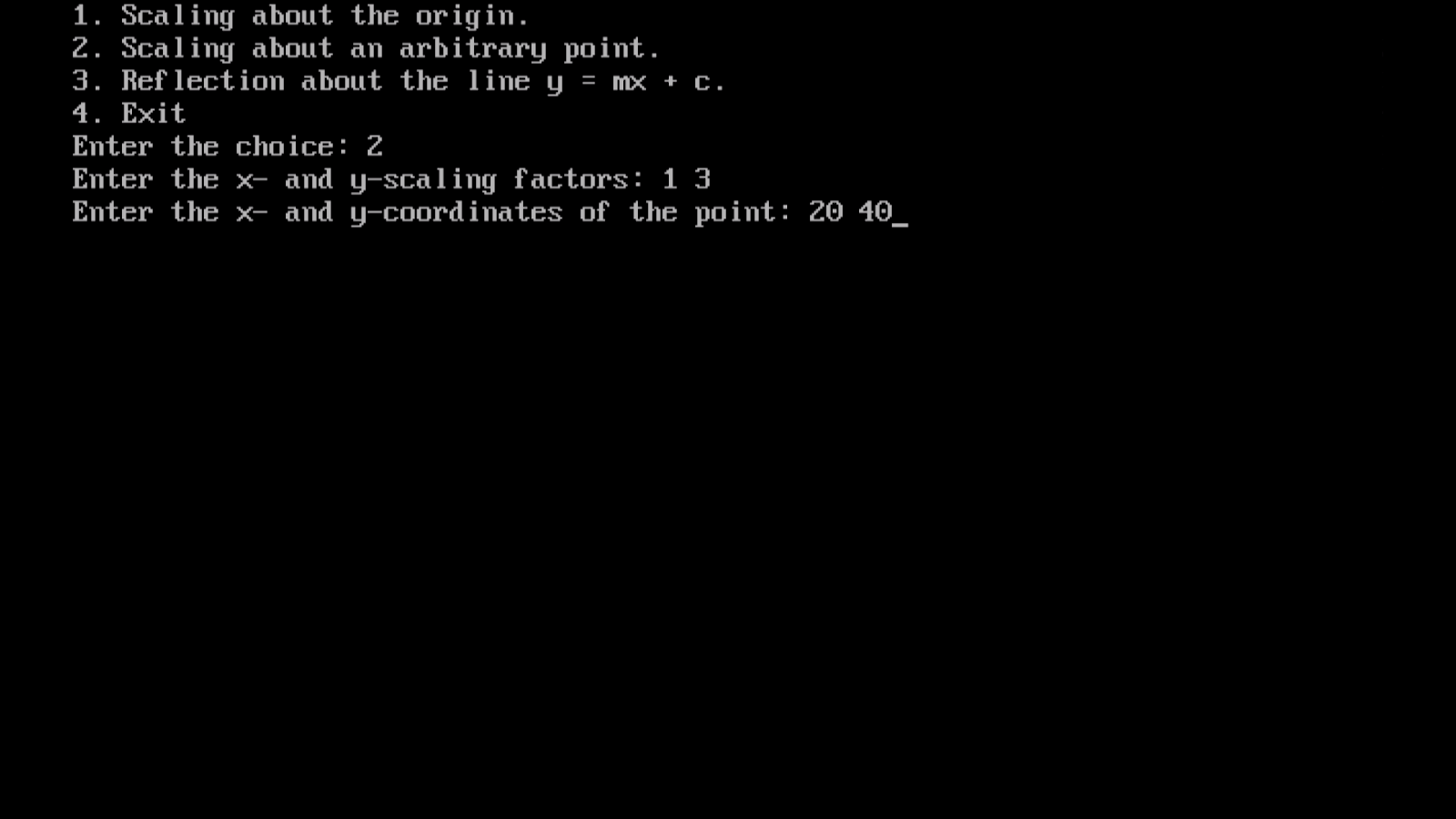
Scaling about Origin:

****

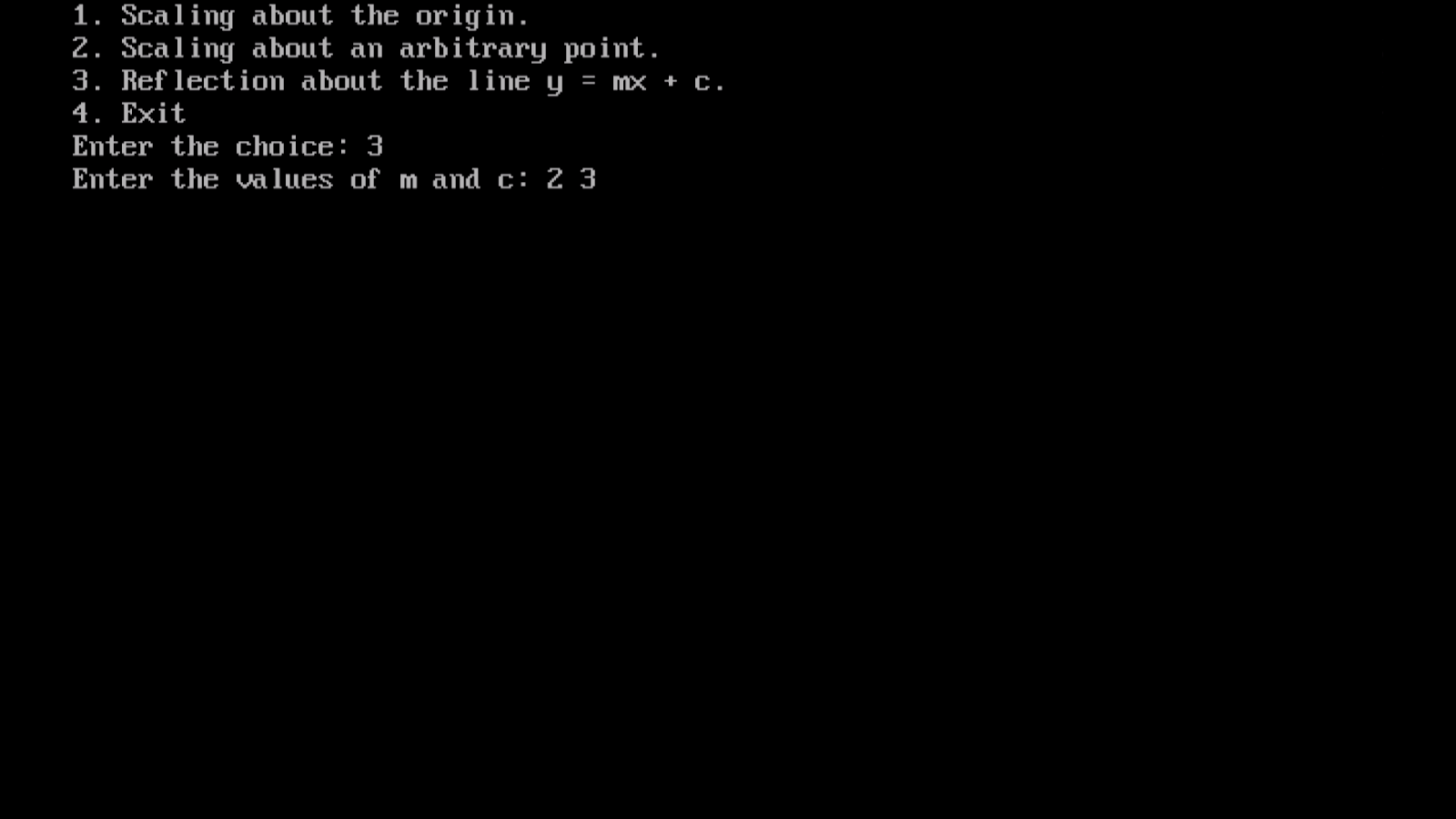
****

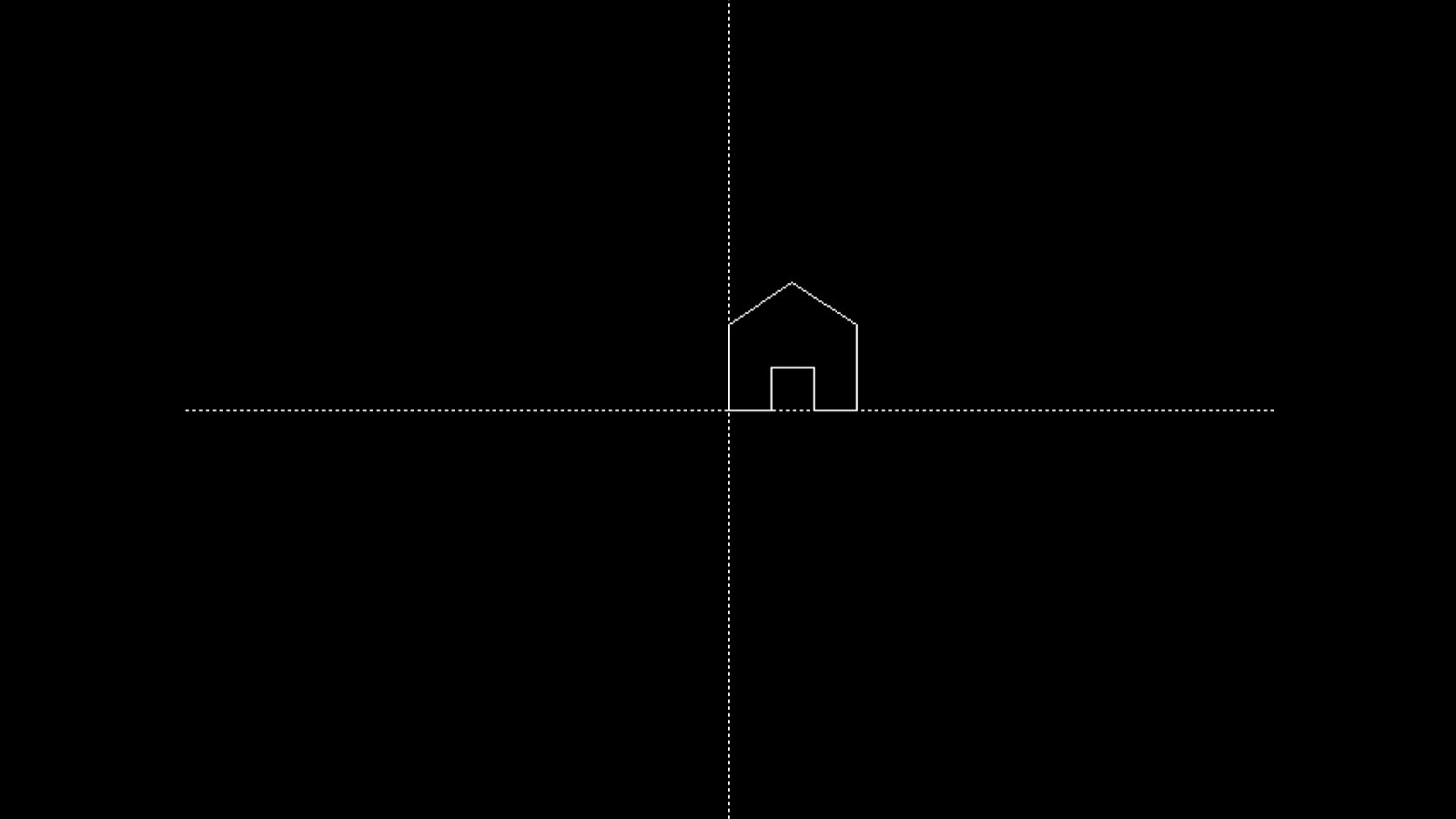
****

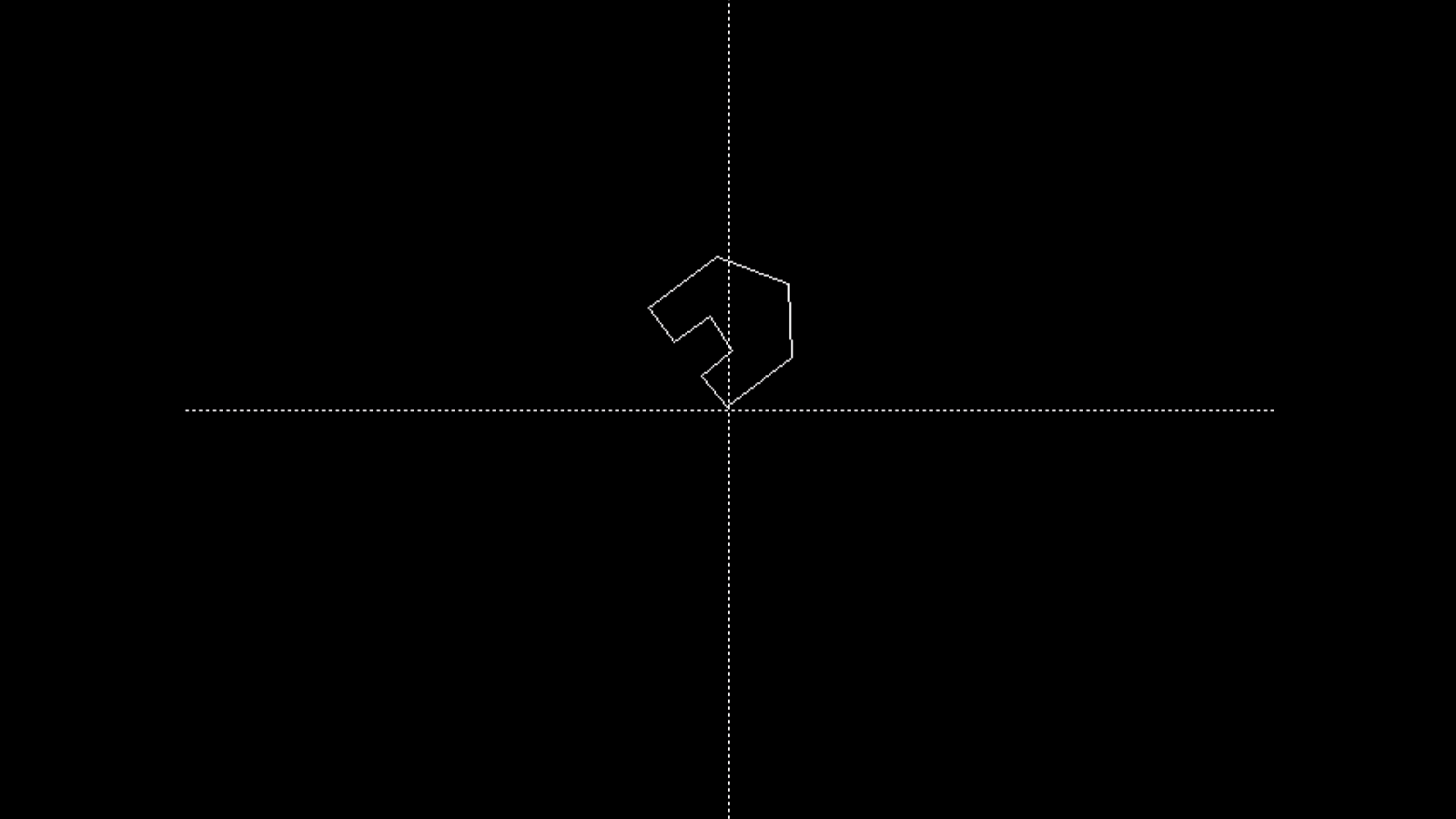
**2.Scaling and translation about arbitrary points**



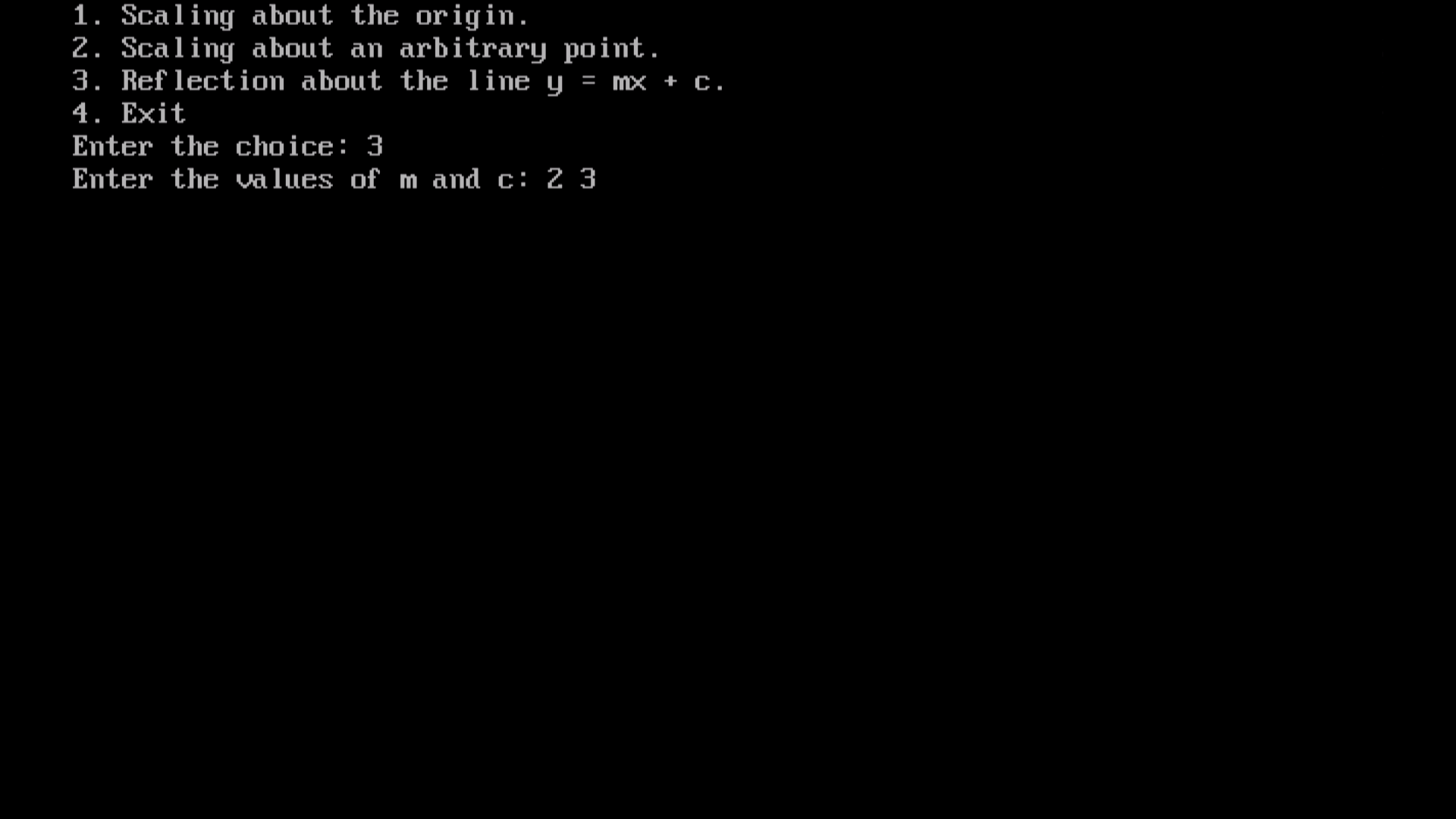
3.Reflection **about the line y = mx + c.**

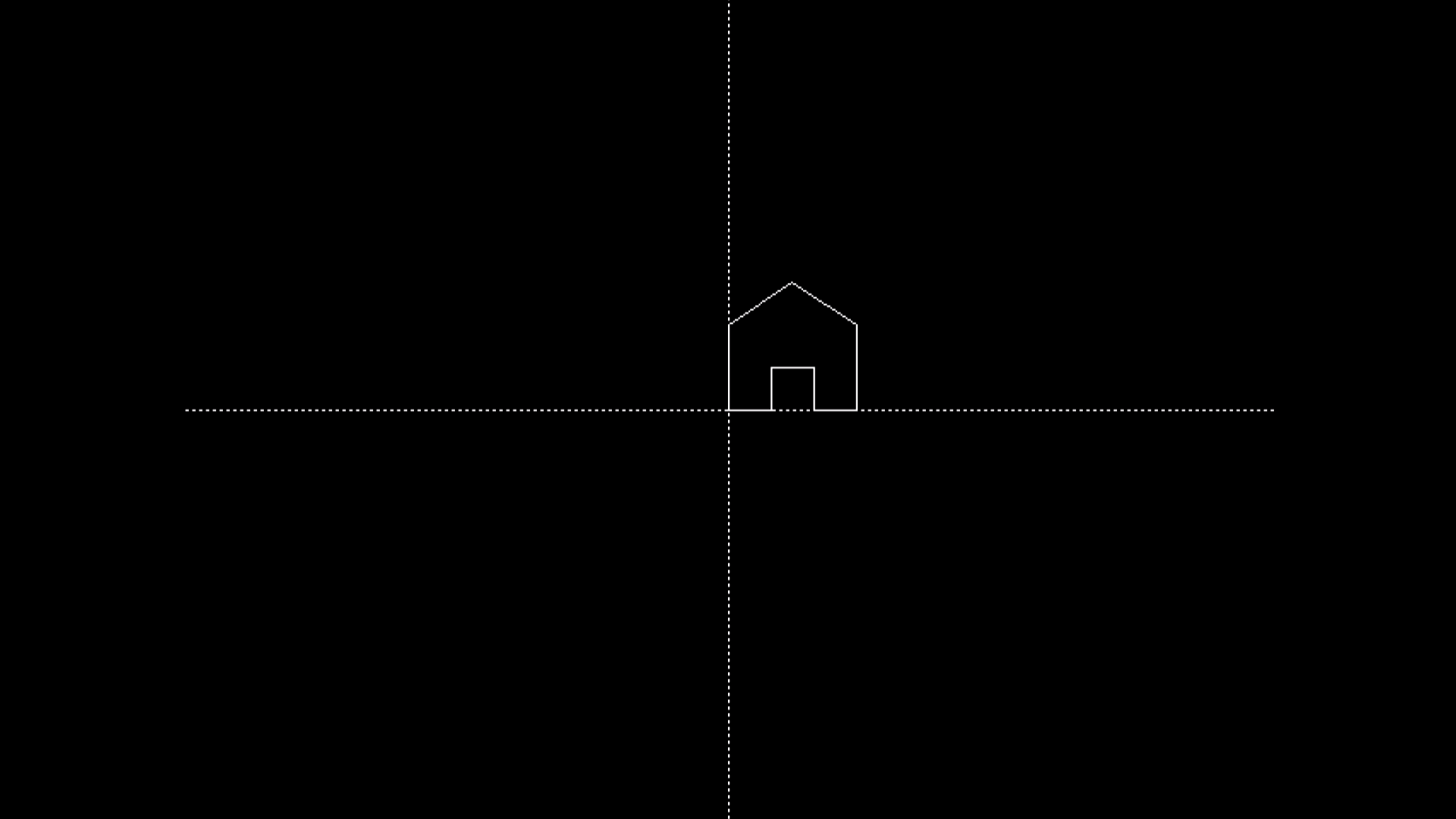
****

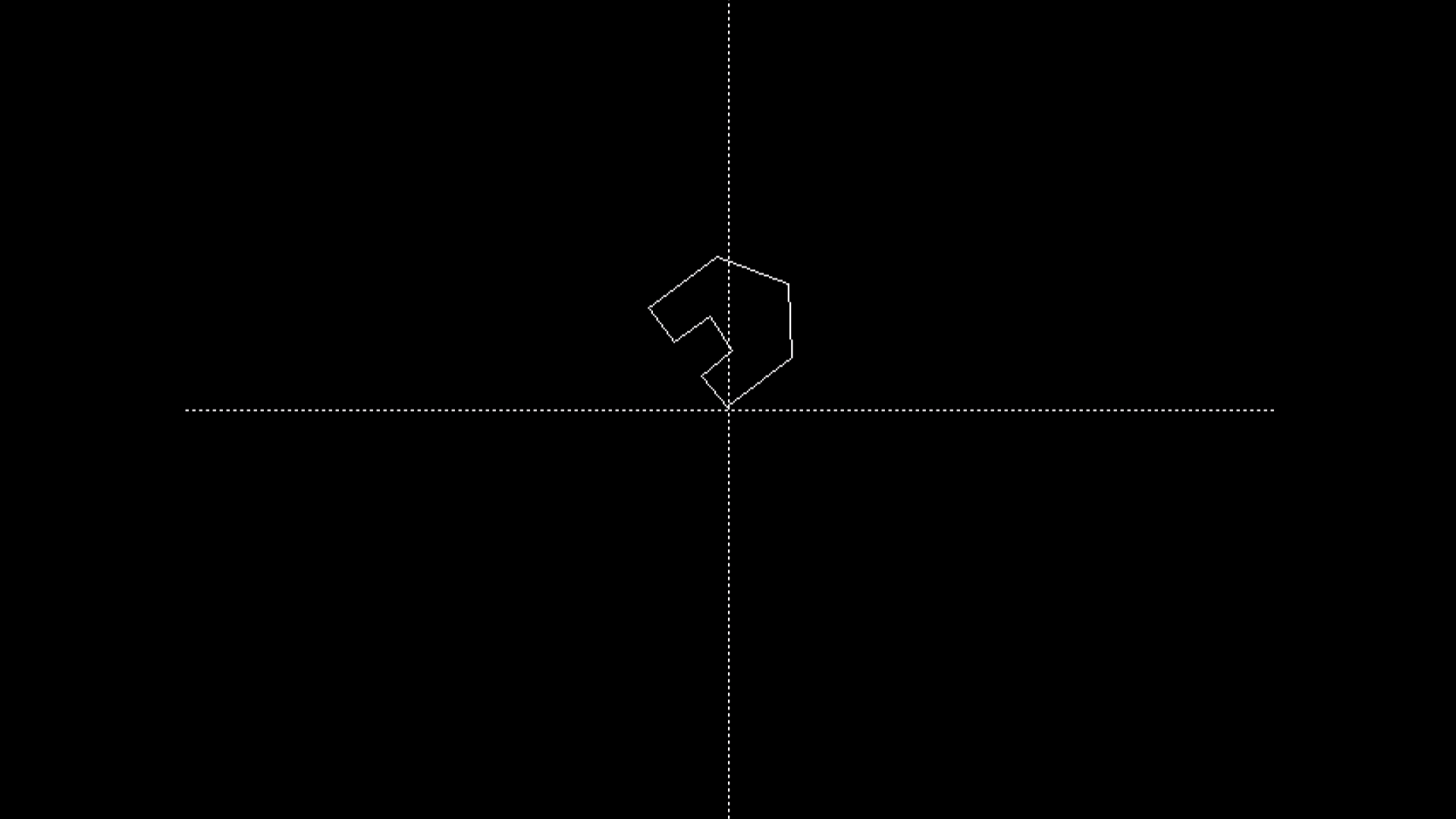
****

****

**3. Reflection about the line y = mx + c.**

****

****

****

Practical 8 - Solve The following

------------------------------------------------------------------------------------------------------

# A.Write a program to implement Cohen-Sutherland clipping.

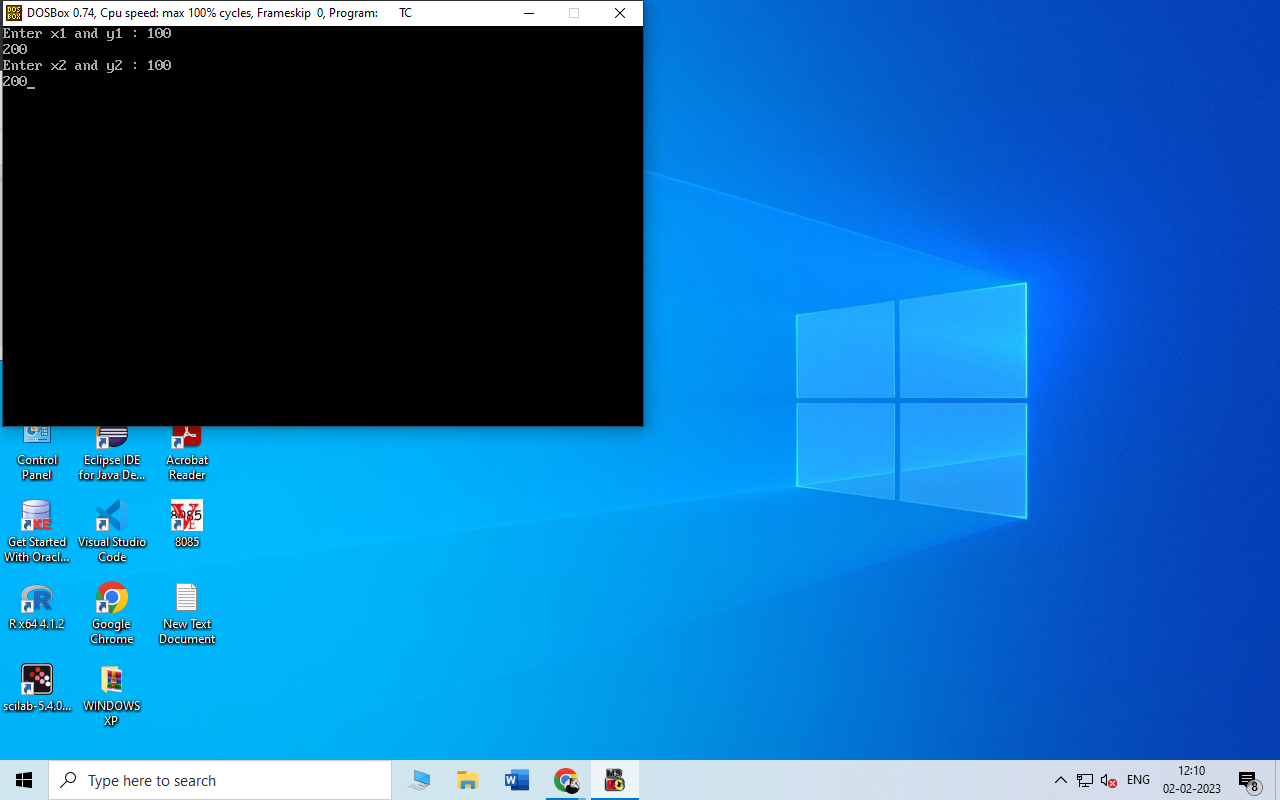
**Source Code in c:**

| #include<stdio.h>  #include<stdlib.h>  #include<math.h>  #include<graphics.h>  #include<dos.h>    typedef struct coordinate  {  int x,y;  char code[4];  }PT;    void drawwindow();  void drawline(PT p1,PT p2);  PT setcode(PT p);  int visibility(PT p1,PT p2);  PT resetendpt(PT p1,PT p2);    void main()  {  int gd=DETECT,v,gm;  PT p1,p2,p3,p4,ptemp;  printf("\nEnter x1 and y1\n");  scanf("%d %d",&p1.x,&p1.y);  printf("\nEnter x2 and y2\n");  scanf("%d %d",&p2.x,&p2.y);  initgraph(&gd,&gm,"c:\\turboc3\\bgi");  drawwindow();  delay(500);  drawline(p1,p2);  delay(500);  cleardevice();  delay(500);  p1=setcode(p1);  p2=setcode(p2);  v=visibility(p1,p2);  delay(500);  switch(v)  {  case 0: drawwindow();  delay(500);  drawline(p1,p2);  break;  case 1: drawwindow();  delay(500);  break;  case 2: p3=resetendpt(p1,p2);  p4=resetendpt(p2,p1);  drawwindow();  delay(500);  drawline(p3,p4);  break;  }  delay(5000);  closegraph();  }    void drawwindow()  {  line(150,100,450,100);  line(450,100,450,350);  line(450,350,150,350);  line(150,350,150,100);  }    void drawline(PT p1,PT p2)  {  line(p1.x,p1.y,p2.x,p2.y);  }    PT setcode(PT p) //for setting the 4 bit code  {  PT ptemp;  if(p.y<100)  ptemp.code[0]='1'; //Top  else  ptemp.code[0]='0';  if(p.y>350)  ptemp.code[1]='1'; //Bottom  else  ptemp.code[1]='0';  if(p.x>450)  ptemp.code[2]='1'; //Right  else  ptemp.code[2]='0';  if(p.x<150)  ptemp.code[3]='1'; //Left  else  ptemp.code[3]='0';  ptemp.x=p.x;  ptemp.y=p.y;  return(ptemp);  }    int visibility(PT p1,PT p2)  {  int i,flag=0;  for(i=0;i<4;i++)  {  if((p1.code[i]!='0') || (p2.code[i]!='0'))  flag=1;  }  if(flag==0)  return(0);  for(i=0;i<4;i++)  {  if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))  flag='0';  }  if(flag==0)  return(1);  return(2);  }    PT resetendpt(PT p1,PT p2)  {  PT temp;  int x,y,i;  float m,k;  if(p1.code[3]=='1')  x=150;  if(p1.code[2]=='1')  x=450;  if((p1.code[3]=='1') || (p1.code[2]=='1'))  {  m=(float)(p2.y-p1.y)/(p2.x-p1.x);  k=(p1.y+(m\*(x-p1.x)));  temp.y=k;  temp.x=x;  for(i=0;i<4;i++)  temp.code[i]=p1.code[i];  if(temp.y<=350 && temp.y>=100)  return (temp);  }  if(p1.code[0]=='1')  y=100;  if(p1.code[1]=='1')  y=350;  if((p1.code[0]=='1') || (p1.code[1]=='1'))  {  m=(float)(p2.y-p1.y)/(p2.x-p1.x);  k=(float)p1.x+(float)(y-p1.y)/m;  temp.x=k;  temp.y=y;  for(i=0;i<4;i++)  temp.code[i]=p1.code[i];  return(temp);  }  else  return(p1);} |
| --- |

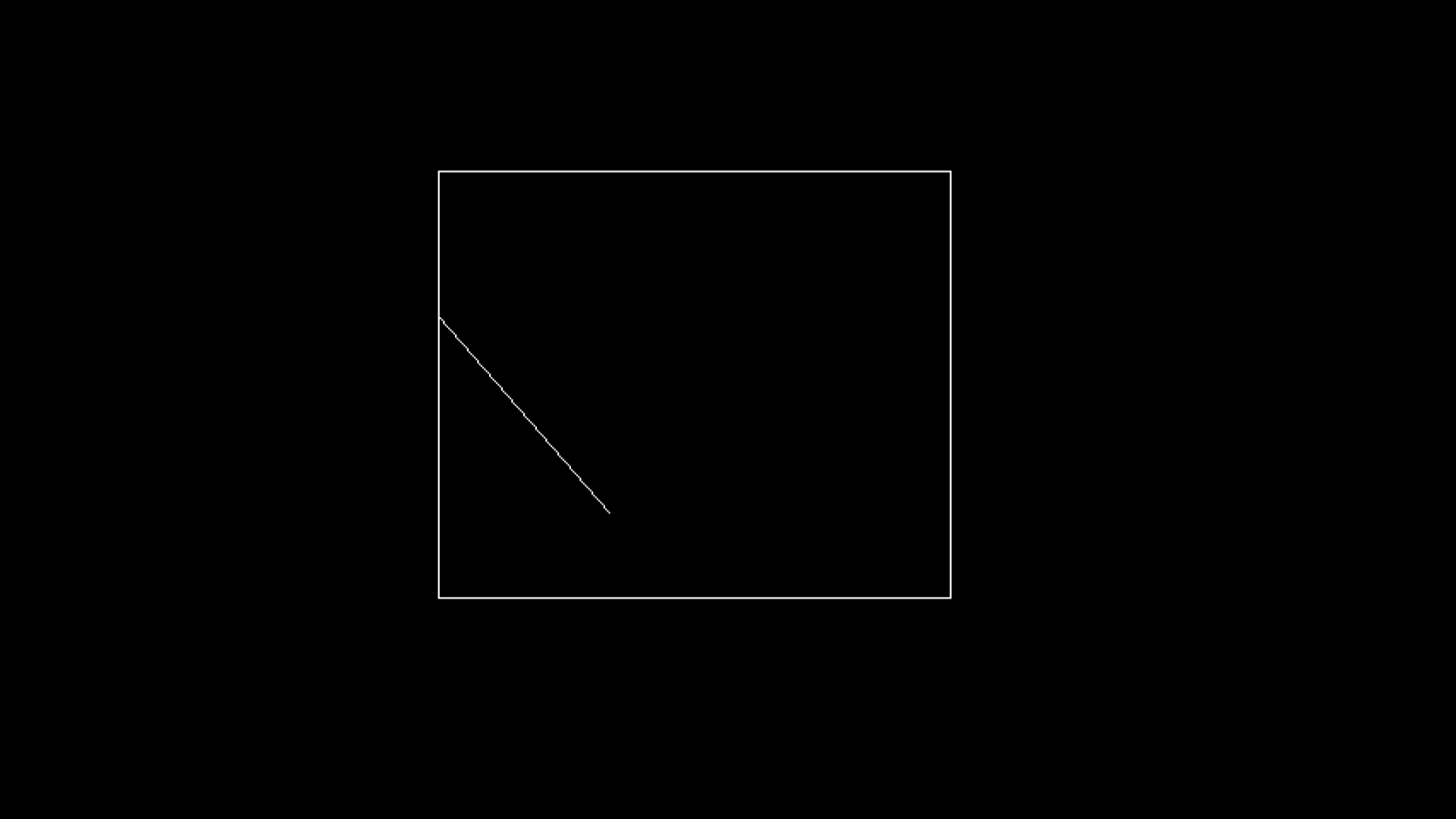
**Source Code in c++:**

| **#include<iostream.h>**  **#include<stdlib.h>**  **#include<math.h>**  **#include<graphics.h>**  **#include<dos.h>**    **typedef struct coordinate**  **{**  **int x,y;**  **char code[4];**  **}PT;**    **void drawwindow();**  **void drawline(PT p1,PT p2);**  **PT setcode(PT p);**  **int visibility(PT p1,PT p2);**  **PT resetendpt(PT p1,PT p2);**    **void main()**  **{**  **int gd=DETECT,v,gm;**  **PT p1,p2,p3,p4,ptemp;**  **cout<<"\nEnter x1 and y1\n";**  **cin>>p1.x>>p1.y;**  **cout<<"\nEnter x2 and y2\n";**  **cin>>p2.x>>p2.y;**  **initgraph(&gd,&gm,"c:\\turboc3\\bgi");**  **drawwindow();**  **delay(500);**  **drawline(p1,p2);**  **delay(500);**  **cleardevice();**  **delay(500);**  **p1=setcode(p1);**  **p2=setcode(p2);**  **v=visibility(p1,p2);**  **delay(500);**  **switch(v)**  **{**  **case 0: drawwindow();**  **delay(500);**  **drawline(p1,p2);**  **break;**  **case 1: drawwindow();**  **delay(500);**  **break;**  **case 2: p3=resetendpt(p1,p2);**  **p4=resetendpt(p2,p1);**  **drawwindow();**  **delay(500);**  **drawline(p3,p4);**  **break;**  **}**  **delay(5000);**  **closegraph();**  **}**    **void drawwindow()**  **{**  **line(150,100,450,100);**  **line(450,100,450,350);**  **line(450,350,150,350);**  **line(150,350,150,100);**  **}**    **void drawline(PT p1,PT p2)**  **{**  **line(p1.x,p1.y,p2.x,p2.y);**  **}**    **PT setcode(PT p) //for setting the 4 bit code**  **{**  **PT ptemp;**  **if(p.y<100)**  **ptemp.code[0]='1'; //Top**  **else**  **ptemp.code[0]='0';**  **if(p.y>350)**  **ptemp.code[1]='1'; //Bottom**  **else**  **ptemp.code[1]='0';**  **if(p.x>450)**  **ptemp.code[2]='1'; //Right**  **else**  **ptemp.code[2]='0';**  **if(p.x<150)**  **ptemp.code[3]='1'; //Left**  **else**  **ptemp.code[3]='0';**  **ptemp.x=p.x;**  **ptemp.y=p.y;**  **return(ptemp);**  **}**    **int visibility(PT p1,PT p2)**  **{**  **int i,flag=0;**  **for(i=0;i<4;i++)**  **{**  **if((p1.code[i]!='0') || (p2.code[i]!='0'))**  **flag=1;**  **}**  **if(flag==0)**  **return(0);**  **for(i=0;i<4;i++)**  **{**  **if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))**  **flag='0';**  **}**  **if(flag==0)**  **return(1);**  **return(2);**  **}**    **PT resetendpt(PT p1,PT p2)**  **{**  **PT temp;**  **int x,y,i;**  **float m,k;**  **if(p1.code[3]=='1')**  **x=150;**  **if(p1.code[2]=='1')**  **x=450;**  **if((p1.code[3]=='1') || (p1.code[2]=='1'))**  **{**  **m=(float)(p2.y-p1.y)/(p2.x-p1.x);**  **k=(p1.y+(m\*(x-p1.x)));**  **temp.y=k;**  **temp.x=x;**  **for(i=0;i<4;i++)**  **temp.code[i]=p1.code[i];**  **if(temp.y<=350 && temp.y>=100)**  **return (temp);**  **}**  **if(p1.code[0]=='1')**  **y=100;**  **if(p1.code[1]=='1')**  **y=350;**  **if((p1.code[0]=='1') || (p1.code[1]=='1'))**  **{**  **m=(float)(p2.y-p1.y)/(p2.x-p1.x);**  **k=(float)p1.x+(float)(y-p1.y)/m;**  **temp.x=k;**  **temp.y=y;**  **for(i=0;i<4;i++)**  **temp.code[i]=p1.code[i];**  **return(temp);**  **}**  **else**  **return(p1);**  **}** |
| --- |

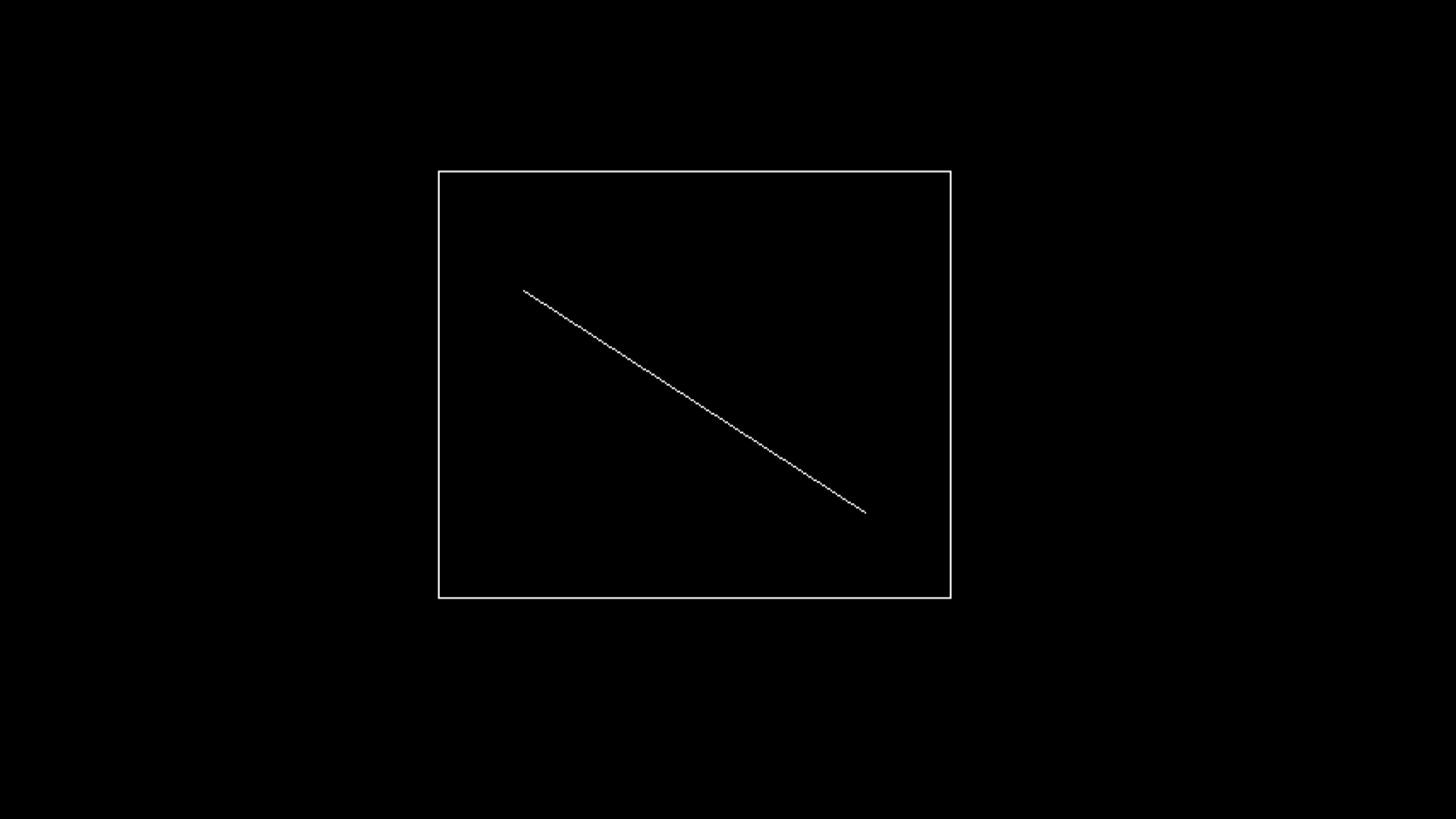
**Output -**



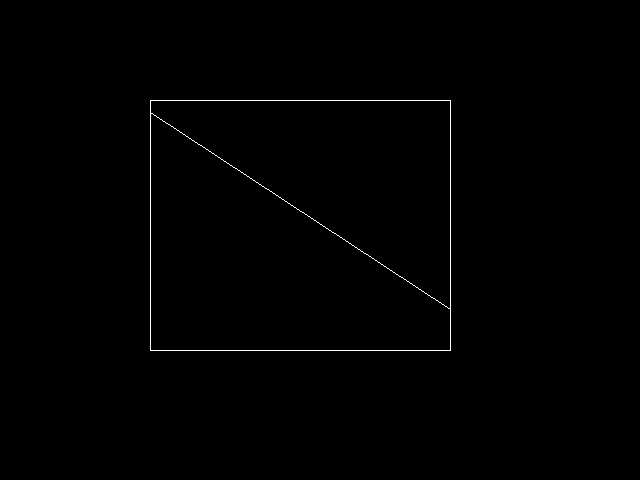
**Line is half in and half out of clipping area:**



**line is Completely Inside of the Clipping Area:**



**Line passes through clipping area:**



Practical 9 - Solve The following

------------------------------------------------------------------------------------------------------

# A.Write a program to fill a circle using Flood Fill Algorithm.

**Source Code -**

| #include<stdio.h>  #include<conio.h>  #include<graphics.h>  #include<dos.h>  void flood(int,int,int,int);  void main()  {  int gd=DETECT,gm;  initgraph(&gd,&gm,"C:/TURBOC3/bgi");  circle(50,50,10);  // rectangle(50,50,250,250);  flood(55,55,14,0);  getch();  }  void flood(int x,int y,int fillColor, int defaultColor)  {  if(getpixel(x,y)==defaultColor)  {  delay(5);  putpixel(x,y,fillColor);  flood(x+1,y,fillColor,defaultColor);  flood(x-1,y,fillColor,defaultColor);  flood(x,y+1,fillColor,defaultColor);  flood(x,y-1,fillColor,defaultColor);  }  } |
| --- |

**Output -**

|  |
| --- |

# 

# 

# B.Write a program to fill a circle using Boundary Fill Algorithm.

**Source Code -**

| #include<iostream.h>  #include<conio.h>  #include<graphics.h>  #include<dos.h>  void boundryfill(int x, int y, int f\_color, int b\_color)  {  if(getpixel(x,y)!= b\_color && getpixel(x,y)!= f\_color)  {  delay(20);  putpixel(x,y,f\_color);  boundryfill(x+1,y,f\_color,b\_color);  boundryfill(x,y+1,f\_color,b\_color);  boundryfill(x-1,y,f\_color,b\_color);  boundryfill(x,y-1,f\_color,b\_color);  }  }  void main()  {  int gd= DETECT, gm,r;  int x,y;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  clrscr();  cout<<"\nEnter x & y position for circle: ";  cin>>x>>y;  cout<<"\nEnter radius of a circle:";  cin>>r;  circle(x,y,r);  boundryfill(x,y,4,15);  getch();  closegraph();  } |
| --- |

**Output -**

|  |
| --- |

Practical 10 - Solve The following

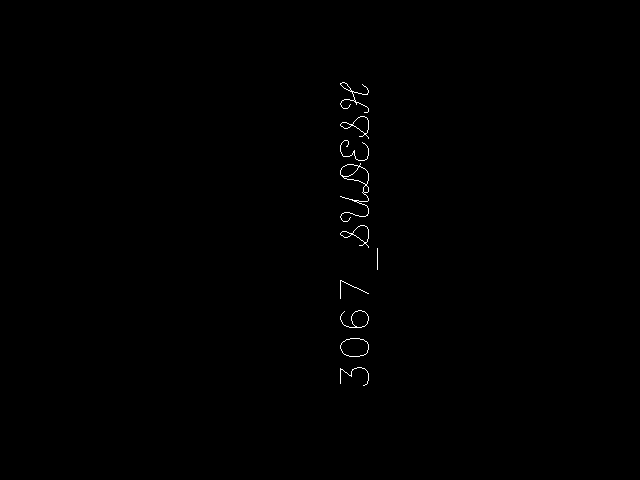
------------------------------------------------------------------------------------------------------

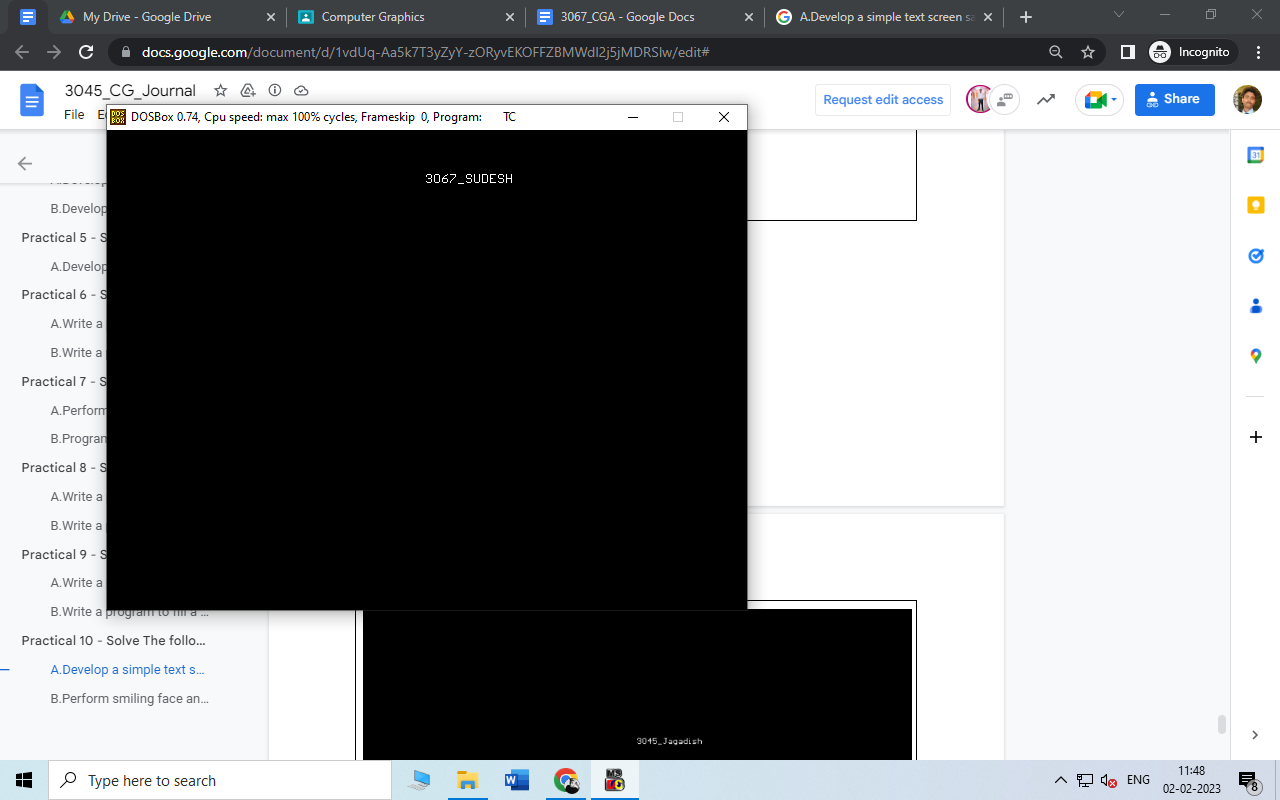
# A.Develop a simple text screen saver using graphics functions.

**Source Code -**

| #include<stdio.h>  #include<conio.h>  #include<dos.h>  #include<graphics.h>  void main()  {  int gd=DETECT,gm,maxx,maxy,key0,i;  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");  maxx=getmaxx();  maxy=getmaxy();  while(!kbhit());  {for(i=0;i<maxy;i++)  {cleardevice();  settextstyle(3,2,5);  outtextxy(maxx/2,i,"3067\_SUDESH");  delay(100);  }}  getch();  } |
| --- |

**Output -**

****

****

# B.Perform smiling face animation using graphic functions.

**Source Code -**

| #include <stdio.h>  #include <conio.h>  #include <graphics.h>  void main()  {  int gd = DETECT, gm;  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");  circle(200,200,30);  circle(190,190,5);  arc(190,190,50,130,10);  circle(210,190,5);  arc(210,190,50,130,10);  arc(200,210,180,360,10);  line(187 , 210 , 193 , 210);  line(207, 210, 213, 210);  line(198, 195, 195, 200);  line(202,195,205,200);  line(195,200,200,205);  line(205,200,200,205);  getch();  closegraph();  } |
| --- |

**Output -**

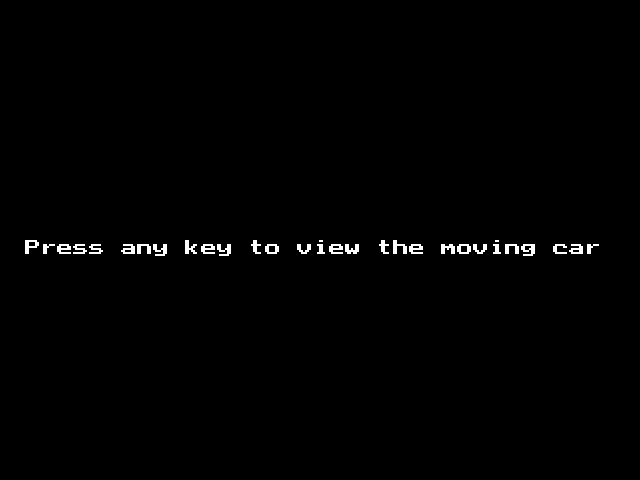
|  |
| --- |

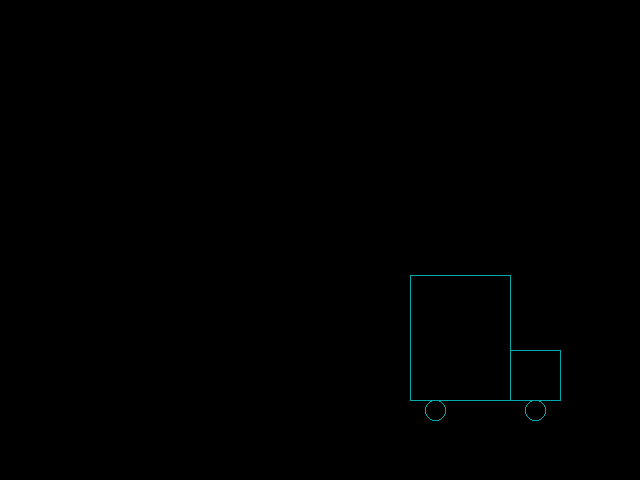
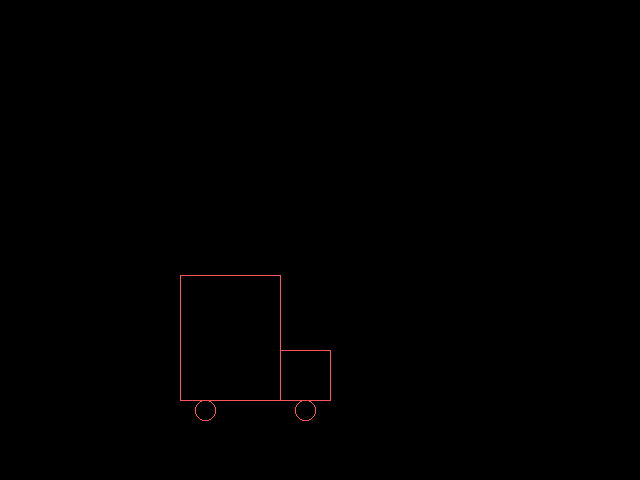
# C. Draw a Moving Car on The Screen

**Source Code:**

| #include<dos.h>  #include<graphics.h>  #include <conio.h>  void main() {  int i,j=0, gd = DETECT , gm;  initgraph (& gd,& gm,"C:\\TURBOC3\\BGI");  settextstyle(DEFAULT\_FONT , HORIZ\_DIR,2);  outtextxy(25,240,"Press any key to view the moving car ");  getch();  setviewport(0,0,639,440,1);  for(i=0;i<=420;i=i+10,j++){  rectangle(50+i,275,150+i,400);  rectangle(150+i,350,200+i,400);  circle(75+i,410,10);  circle(175+i,410,10);  setcolor(j);  delay(100);  if(i==420)  break;  clearviewport();  }  getch();  closegraph();  } |
| --- |

**Output:**

****

****