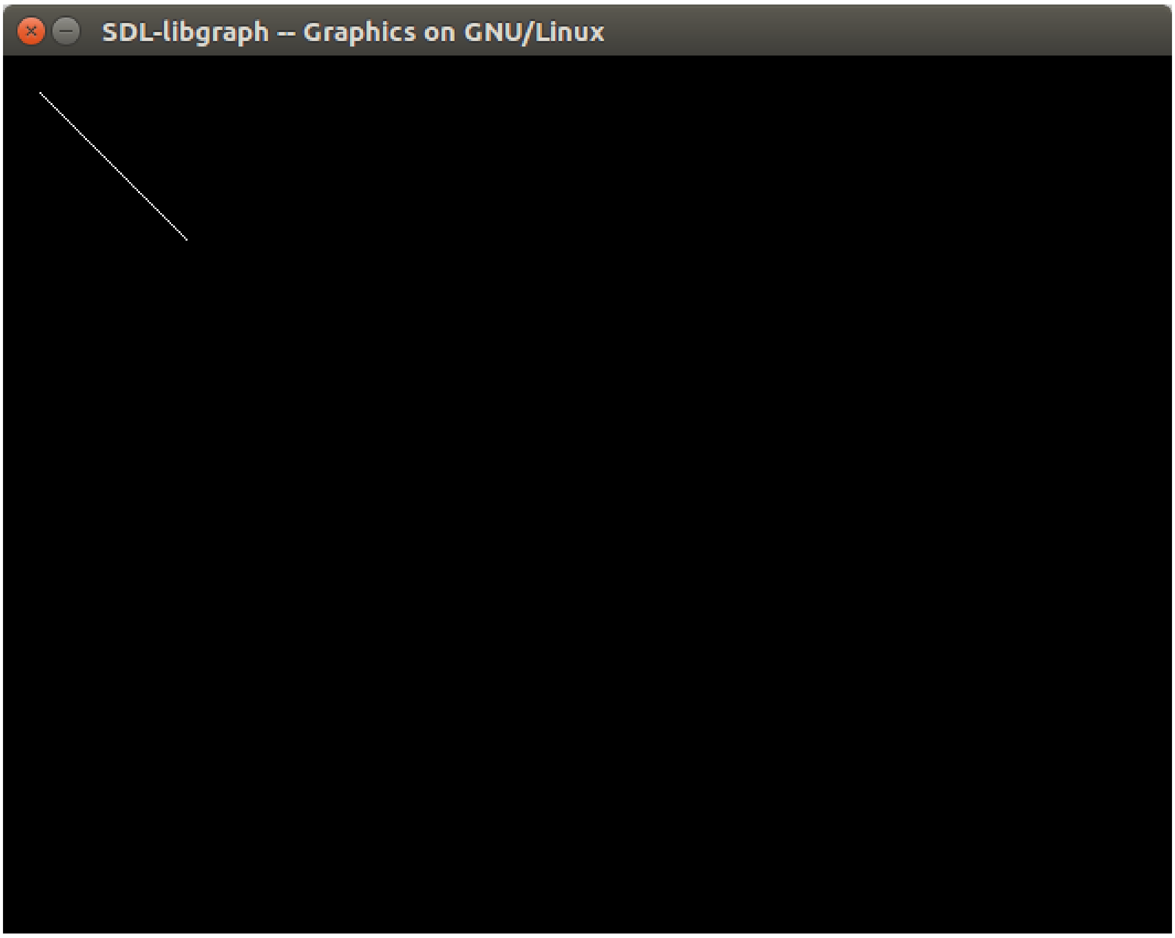
Practical 1

Task 1 : Write a program to draw a line using the built-in line function of “Graphics.h” .

Source Code:

```
#include<graphics.h>
int main()
{
        int gd = DETECT,gm;
        initgraph(&gd,&gm,NULL);
        line(20,20,100,100);
        delay(5000);
        closegraph();
        return 0;
}
```

Output:

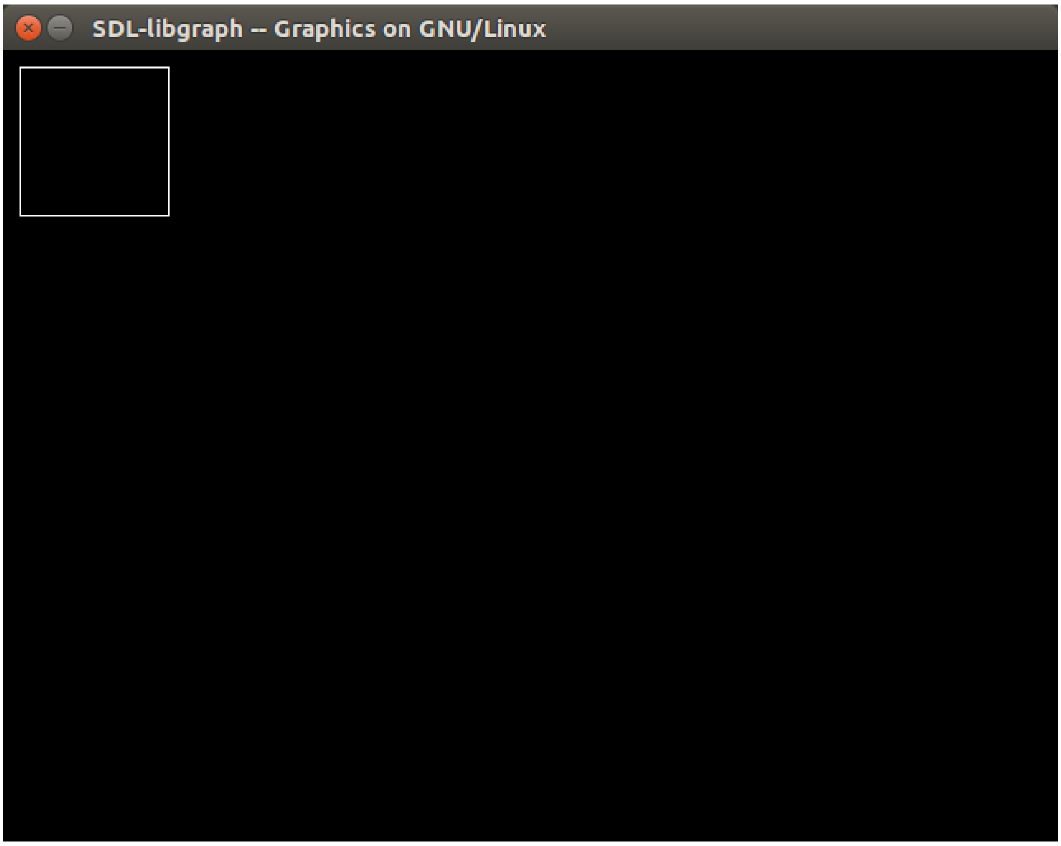


Task 2 : Write a program to draw a rectangle using the built-in line function of "Graphics.h" .

Source Code:

```
#include<graphics.h>
int main()
{
  int gd = DETECT,gm,left=100,top=100,right=200,bottom=200,x=
300,y=150,radius=50;
  initgraph(&gd,&gm,NULL);
  line(10,10,100,10);
  line(10,10,10,100);
  line(10,100,100,100);
  line(100,10,100,100);

  delay(5000);
  closegraph();
  return 0;
}
```

Output :



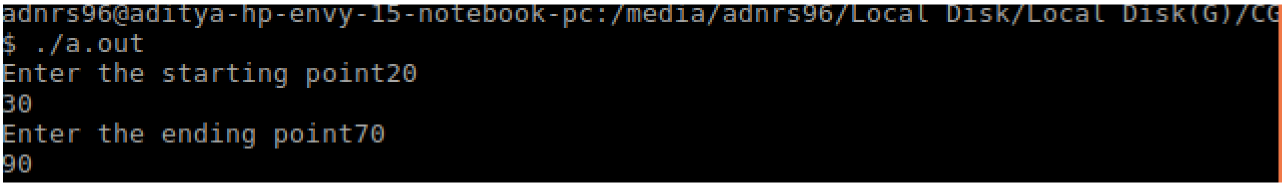
Practical 2

Task 1 : Write a program to draw a line using DDA-Line Drawing algorithm and “Graphics.h”.

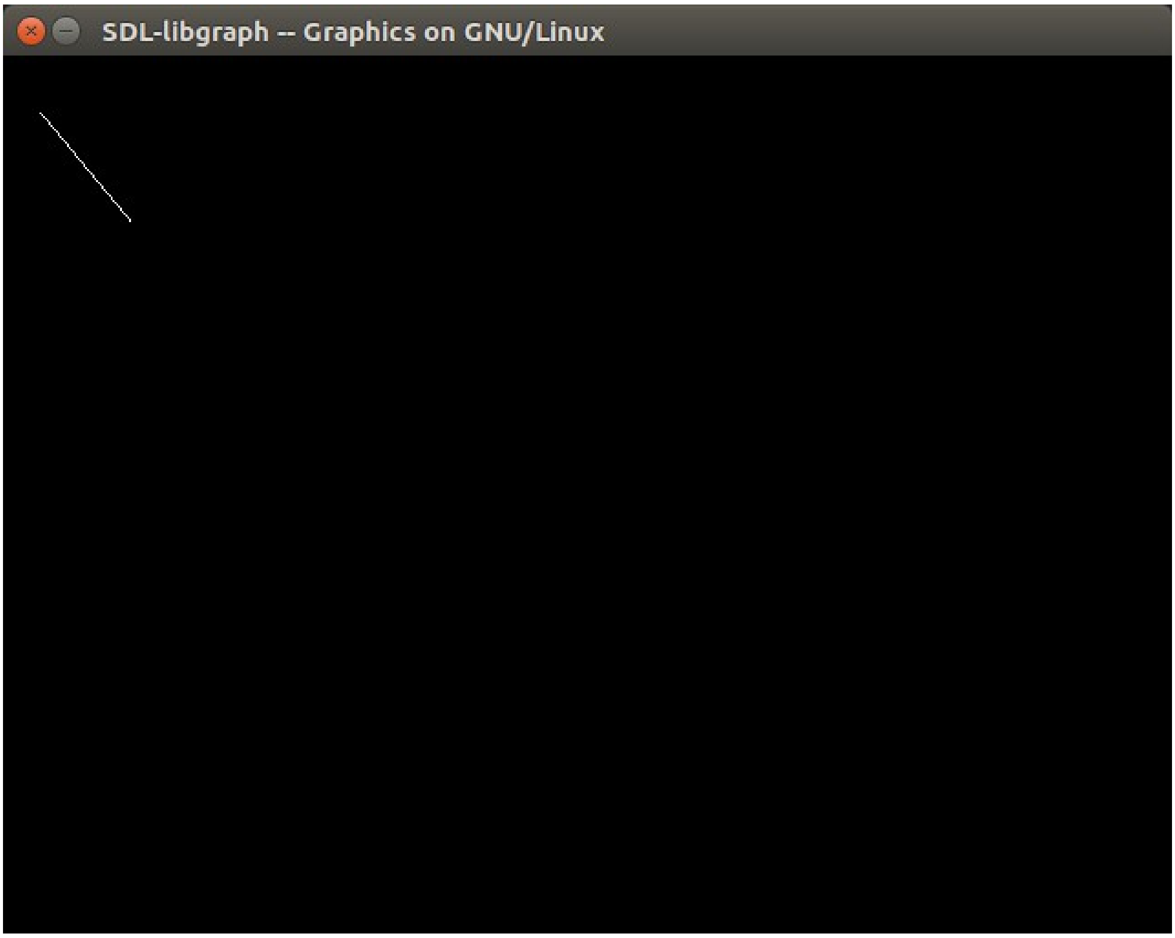
Source Code:

```
#include<graphics.h>
#include<stdio.h>
int main()
{
  int gd = DETECT,gm;
  int xa,ya,xb,yb;
  float xi,yi,steps;
  printf("Enter the starting point");
  scanf("%d %d",&xa,&ya);
  printf("Enter the ending point");
  scanf("%d %d",&xb,&yb);
  initgraph(&gd,&gm,NULL);

  int dx = xb-xa;
  int dy = yb-ya;
  if(dy<dx)
     {
        steps=dx;
     }
  else
  {
     steps=dy;
  }
  xi=(float)dx/steps;
  yi=(float)dy/steps;
  float x=xa,y=ya;
  int k=0;
  while(k<steps){
     x+=xi;y+=yi;
     putpixel(x,y,WHITE);
     k++;
  }
  delay(5000);
  closegraph();
  return 0;}
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the starting point20
30
Enter the ending point70
90
```



Practical 3

Task 1 : Write a program to draw a line using Bresenham’s Line Drawing algorithm and “Graphics.h”.

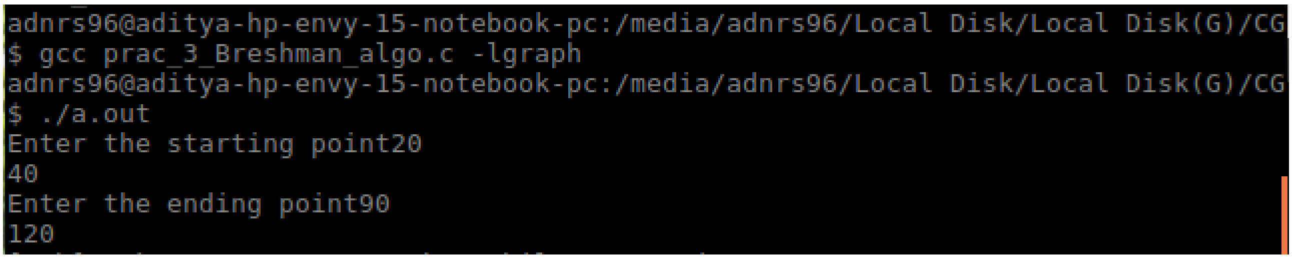
Source Code:

```
#include<graphics.h>
#include<stdio.h>

int main()
{
   int gd = DETECT,gm;
   int xa,ya,xb,yb;
   float xi,yi,steps;
   printf("Enter the starting point");
   scanf("%d %d",&xa,&ya);
   printf("Enter the ending point");
   scanf("%d %d",&xb,&yb);
   initgraph(&gd,&gm,NULL);
   if((xb<xa && yb<ya)||(xb>xa && yb>ya))
   {
     if (xb<xa && yb<ya)
     {
       int temp=xb;
       xb=xa;
       xa=temp;
       temp=yb;
       yb=ya;
       ya=temp;
     }
     int dx = xb-xa;
     int dy = yb-ya;
     int D = dy-dx;
     int y = ya;
     for ( int x = xa;x<xb;x++)
     {
       putpixel(x,y,WHITE);
       if ( D>=0)
       {
         y++;
         D=D-dx;
```

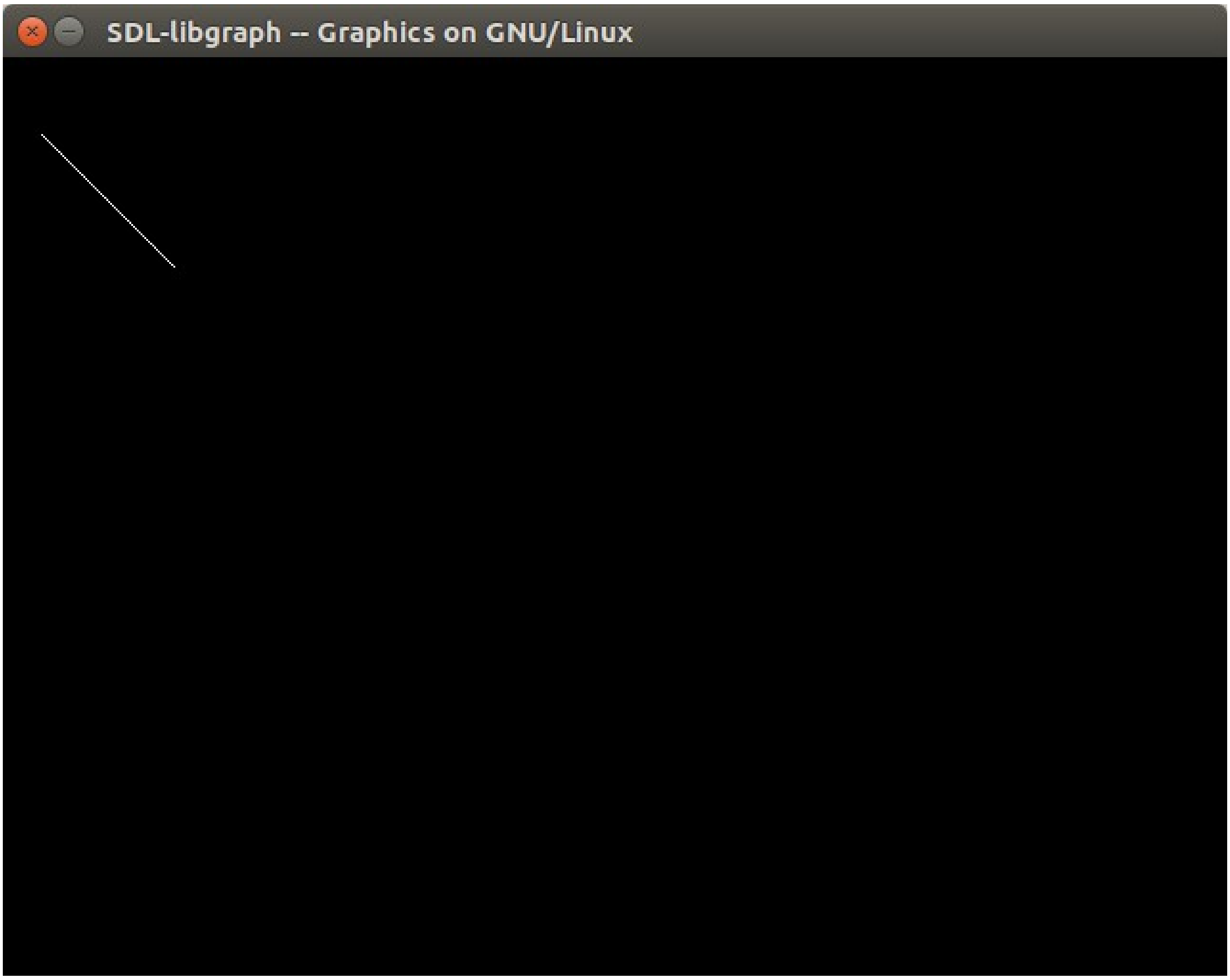
```
      }
      D=D+dy;
    }
  }
  else
  {
    if (xb<xa && yb>ya)
    {
      int temp=xb;
      xb=xa;
      xa=temp;
      temp=yb;
      yb=ya;
      ya=temp;
    }
    int dx = xb-xa;
    int dy = yb-ya;
    int D = dy-dx;
    int x = xa;
    for ( int y = ya;y>yb;y--)
    {
      putpixel(x,y,WHITE);
      if ( D>=0)
      {
        x++;
        D=D-dy;
      }
      D=D+dx;
    }
  }
  delay(5000);
  closegraph();
  return 0;
}
```

Output:

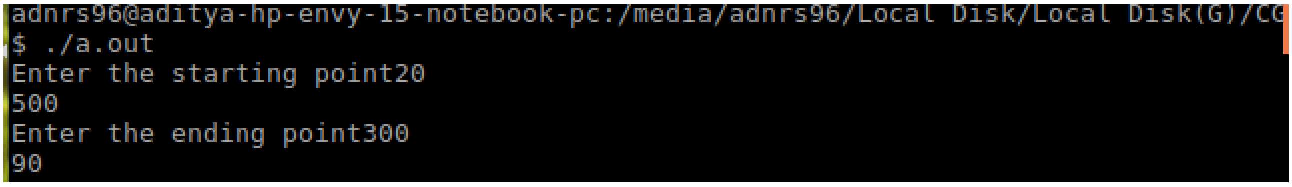
For Positive Slope:



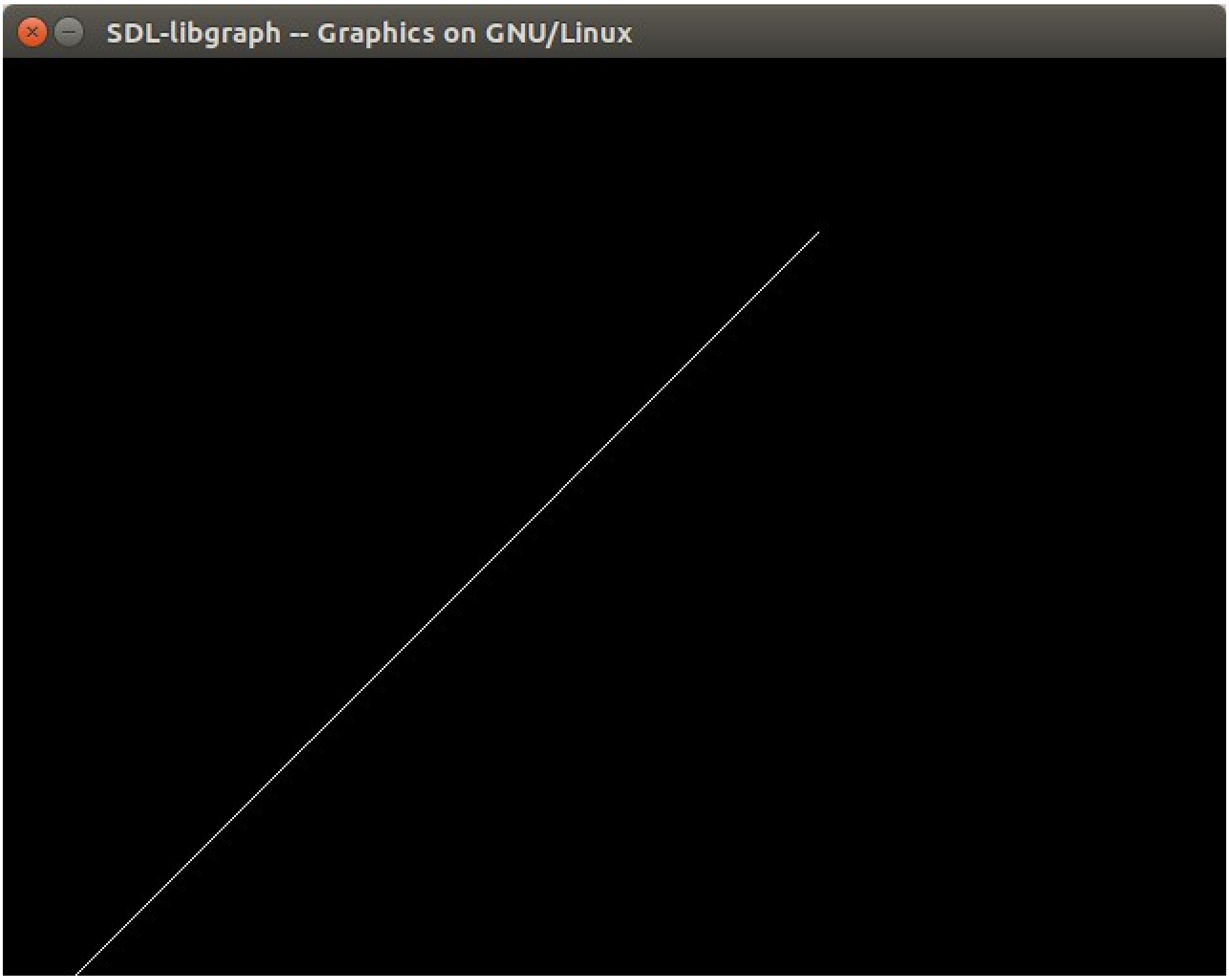
```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ gcc prac_3_Breshman_algo.c -lgraph
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the starting point20
40
Enter the ending point90
120
```



For Negative Slope:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the starting point20
500
Enter the ending point300
90
```



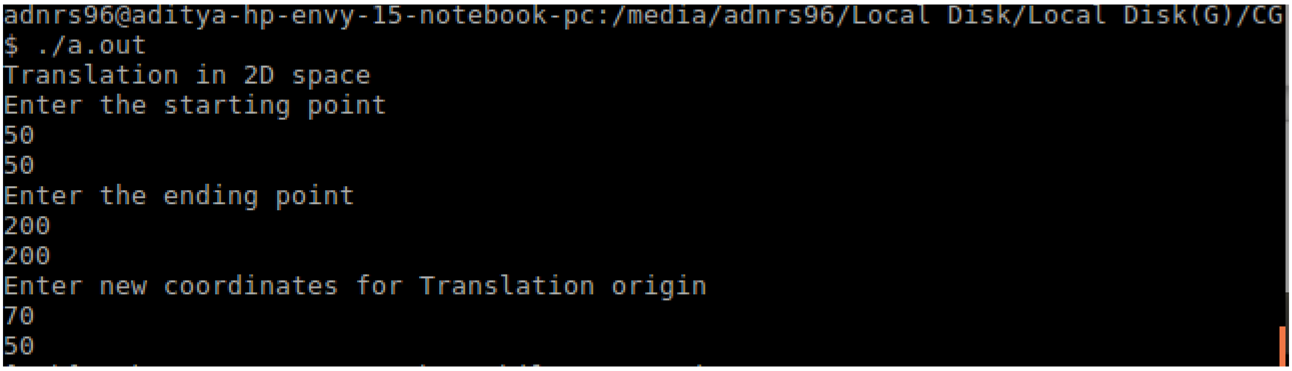
Practical 4

Task 1 : Write a program to perform 2D-Translation operations.

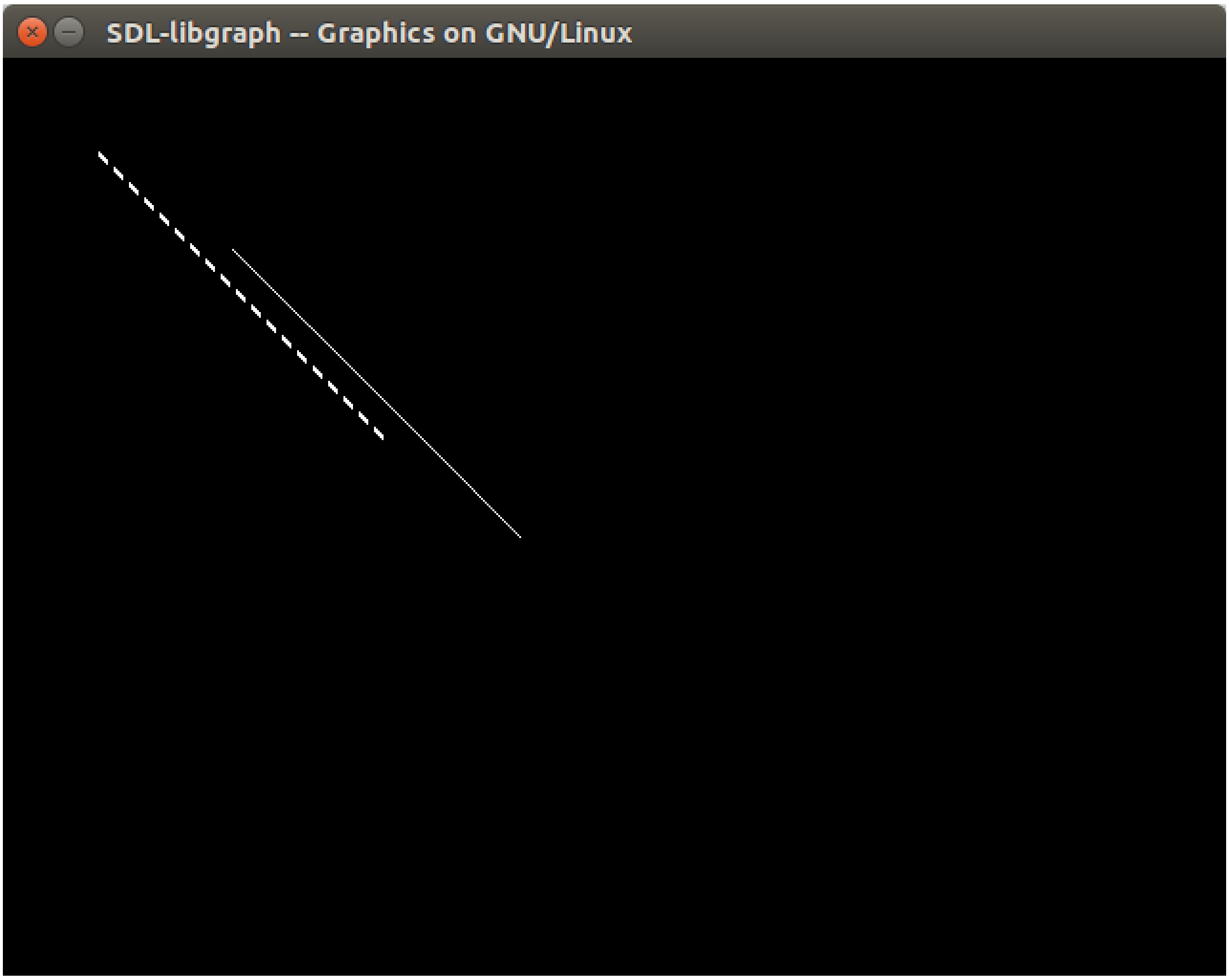
Source Code:

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>

#define PI 3.14159265

int main()
{
  int gd = DETECT,gm;

  float xa,ya,xb,yb,xao,yao,xbo,ybo;

  printf("Translation in 2D space\n");
  printf("Enter the starting point\n");
  scanf("%f %f",&xa,&ya);
  printf("Enter the ending point\n");
  scanf("%f %f",&xb,&yb);
  xao=xa,yao=ya,xbo=xb,ybo=yb;
  int ox,oy;
  printf("Enter new coordinates for Translation origin\n");
  scanf("%d %d",&ox,&oy);
  xa=xa+ox;
  xb=xb+ox;
  ya=ya+oy;
  yb=yb+oy;

  initgraph(&gd,&gm,NULL);
  line(xa,ya,xb,yb);
  setlinestyle(DASHED_LINE,0,THICK_WIDTH);
  line(xao,yao,xbo,ybo);
  delay(5000);
  closegraph();
  return 0;
}
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Translation in 2D space
Enter the starting point
50
50
Enter the ending point
200
200
Enter new coordinates for Translation origin
70
50
```



DASHED_LINE is the original line.

Practical 5

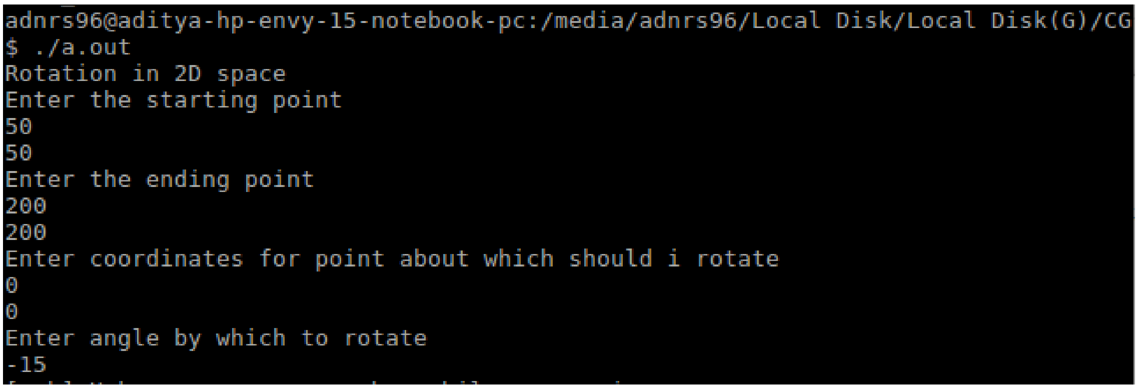
Task 1 : Write a program to perform 2D-Rotation operations.

Source Code:

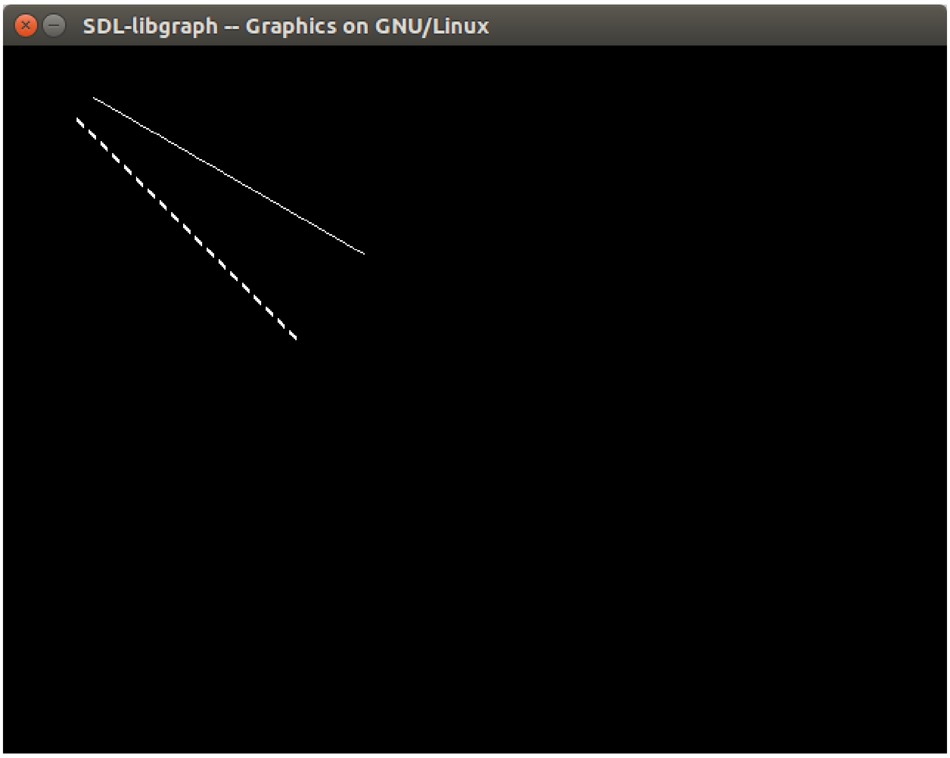
```
#include<graphics.h>
#include<stdio.h>
#include<math.h>

#define PI 3.14159265

int main()
{
  int gd = DETECT,gm;
  float xa,ya,xb,yb,xao,yao,xbo,ybo;
  printf("Rotation in 2D space\n");
  printf("Enter the starting point\n");
  scanf("%f %f",&xa,&ya);
  printf("Enter the ending point\n");
  scanf("%f %f",&xb,&yb);
  xao=xa,yao=ya,xbo=xb,ybo=yb;
  int rx,ry;
  double ang,val;
  printf("Enter coordinates for point about which should i rotate\n");
  scanf("%d %d",&rx,&ry);
  printf("Enter angle by which to rotate\n");
  scanf("%lf",&ang);
  xa=xa+rx;
  xb=xb+rx;
  ya=ya+ry;
  yb=yb+ry;
  val = PI / 180.0;
  ang=ang*val;
  float nxa=xa,nya=ya,nyb=yb,nxb=xb;
  xa = (nxa*cos(ang))-(nya*sin(ang));
  ya = (nxa*sin(ang))+(nya*cos(ang));
  xb = (nxb*cos(ang))-(nyb*sin(ang));
  yb = (nxb*sin(ang))+(nyb*cos(ang));
  xa=xa-rx;
  xb=xb-rx;
  ya=ya-ry;
```

```
    yb=yb-ry;

    initgraph(&gd,&gm,NULL);
    line(xa,ya,xb,yb);
    setlinestyle(DASHED_LINE,0,THICK_WIDTH);
    line(xao,yao,xbo,ybo);
    delay(5000);
    closegraph();
    return 0;
  }
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Rotation in 2D space
Enter the starting point
50
50
Enter the ending point
200
200
Enter coordinates for point about which should i rotate
0
0
Enter angle by which to rotate
-15
```



DASHED_LINE is the original line.

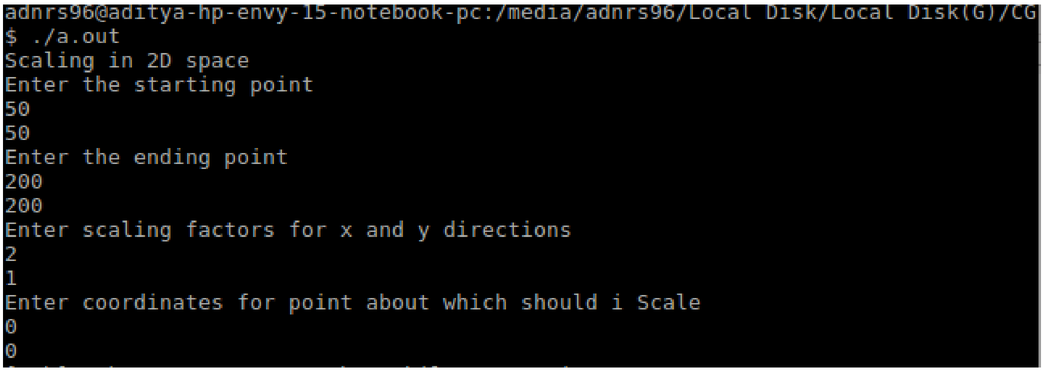
Practical 6

Task 1 : Write a program to perform 2D-Scaling operations.

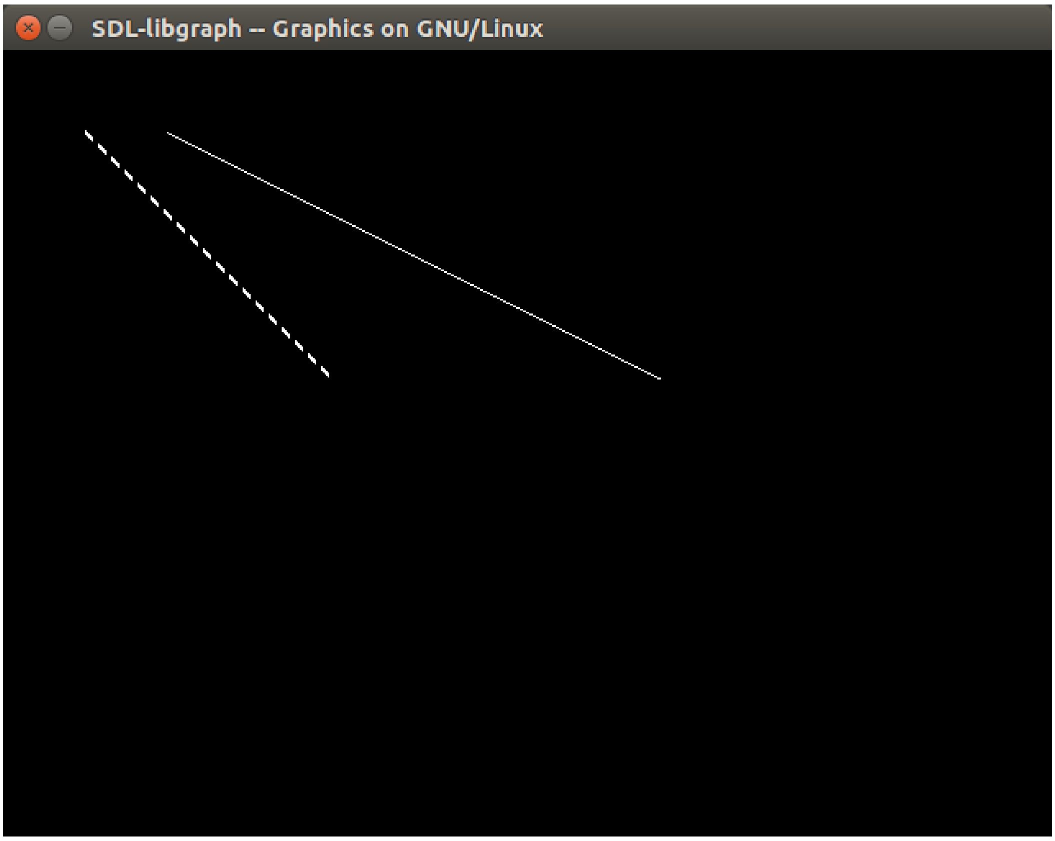
Source Code:

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>

#define PI 3.14159265

int main()
{
  int gd = DETECT,gm;
  float xa,ya,xb,yb,xao,yao,xbo,ybo;
  printf("Scaling in 2D space\n");
  printf("Enter the starting point\n");
  scanf("%f %f",&xa,&ya);
  printf("Enter the ending point\n");
  scanf("%f %f",&xb,&yb);
  xao=xa,yao=ya,xbo=xb,ybo=yb;
  int sx,sy;
  printf("Enter scaling factors for x and y directions\n");
  scanf("%d %d",&sx,&sy);
  int rx,ry;
  printf("Enter coordinates for point about which should i Scale\n");
  scanf("%d %d",&rx,&ry);
  xa=xa+rx;
  xb=xb+rx;
  ya=ya+ry;
  yb=yb+ry;

  xa=xa*sx;
  xb=xb*sx;
  ya=ya*sy;
  yb=yb*sy;

  xa=xa-rx;
  xb=xb-rx;
  ya=ya-ry;
  yb=yb-ry;
  initgraph(&gd,&gm,NULL);
  line(xa,ya,xb,yb);
  setlinestyle(DASHED_LINE,0,THICK_WIDTH);
  line(xao,yao,xbo,ybo);
  delay(5000);
  closegraph();
  return 0;
}
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Scaling in 2D space
Enter the starting point
50
50
Enter the ending point
200
200
Enter scaling factors for x and y directions
2
1
Enter coordinates for point about which should i Scale
0
0
```



DASHED_LINE is the original line.

Practical 7

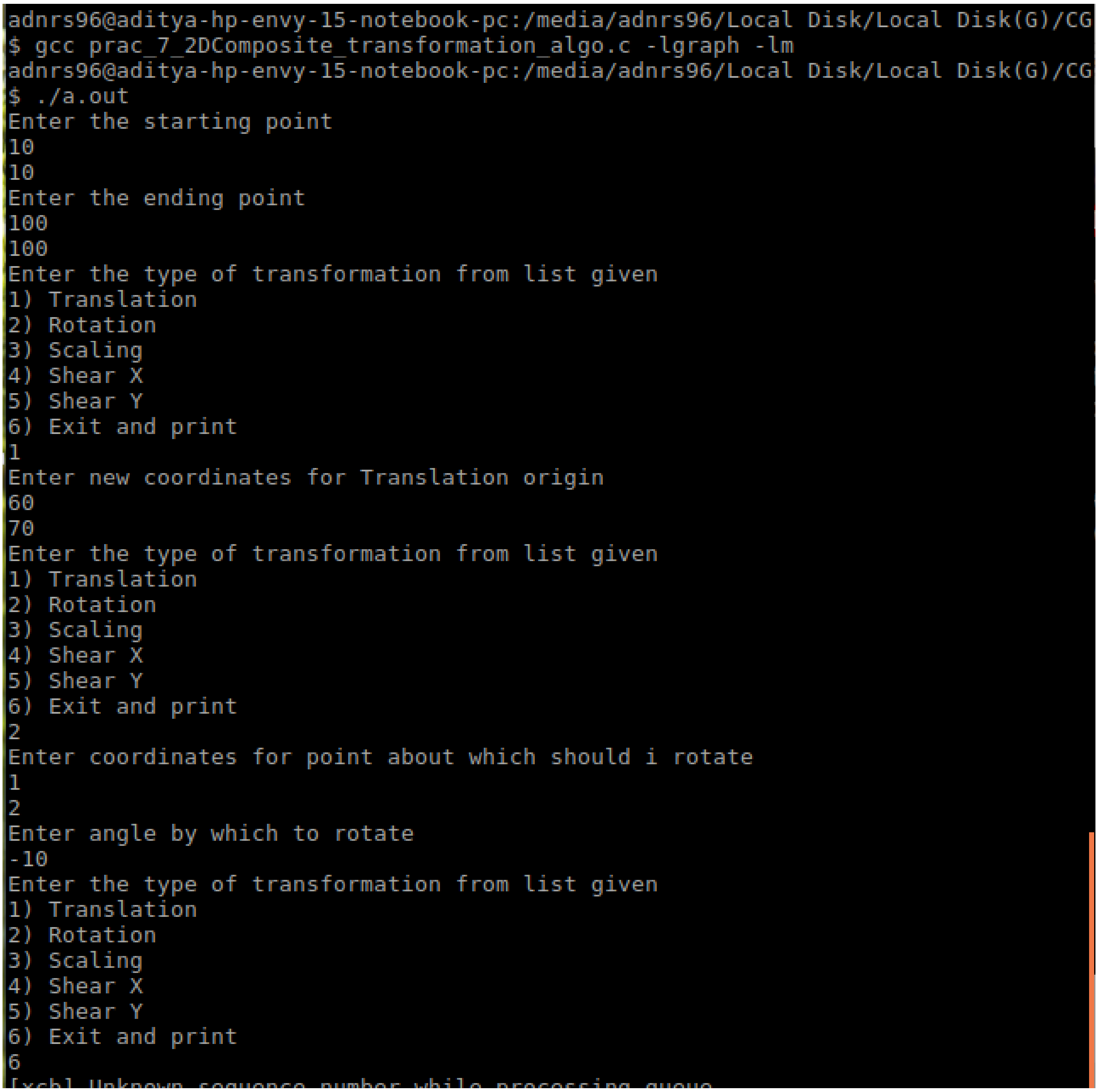
Task 1 : Write a program to perform Composite 2D-Transformation operations.

Source Code:

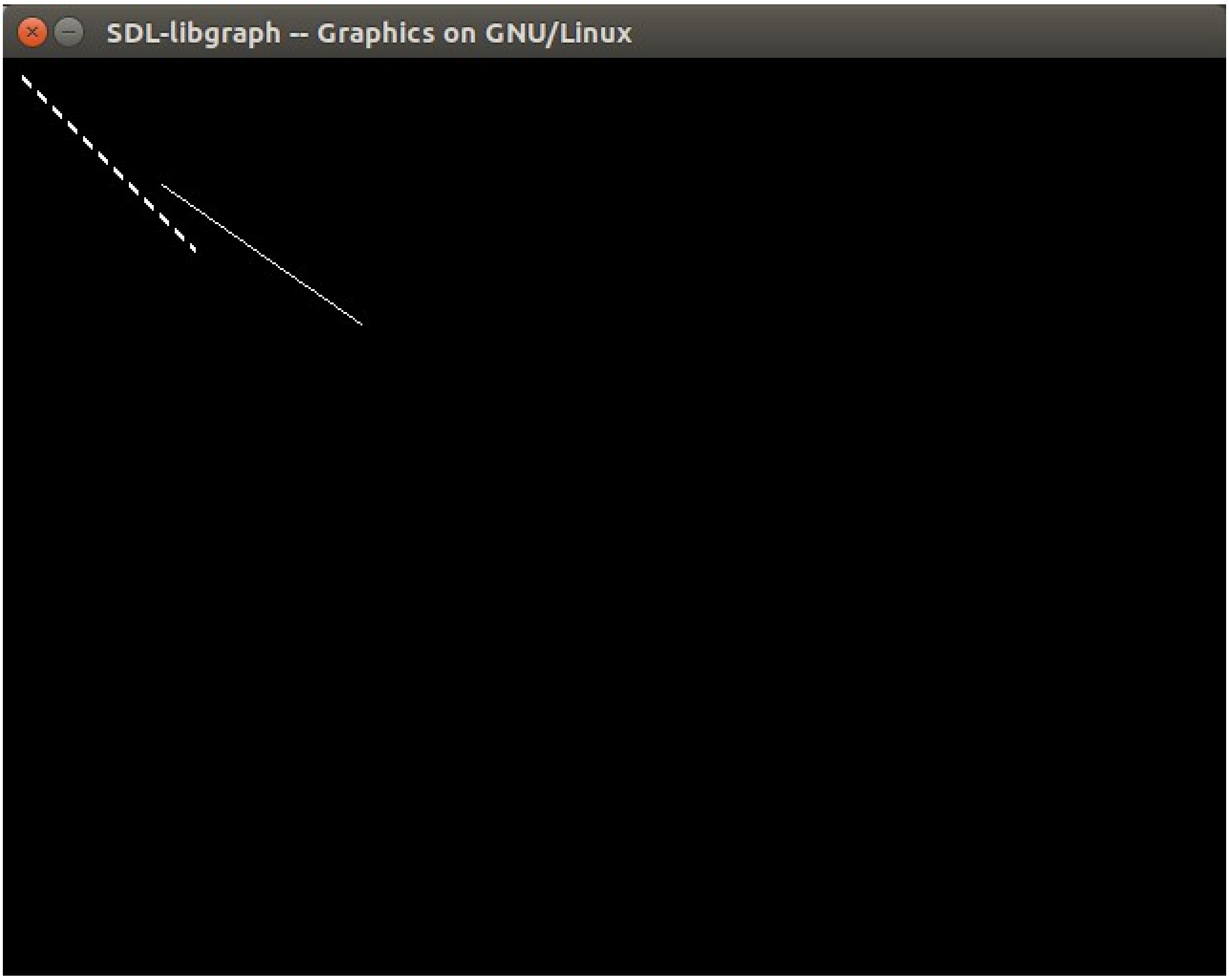
```
#include<graphics.h>
#include<stdio.h>
#include<math.h>

#define PI 3.14159265

int main()
{
   int gd = DETECT,gm;
   float xa,ya,xb,yb,xao,yao,xbo,ybo;

   printf("Enter the starting point\n");
   scanf("%f %f",&xa,&ya);
   printf("Enter the ending point\n");
   scanf("%f %f",&xb,&yb);
   xao=xa,yao=ya,xbo=xb,ybo=yb;

   int choice=0;
   while(choice!=6){
     choice=0;
   printf("Enter the type of transformation from list given\n");
   printf("1) Translation\n");
   printf("2) Rotation\n");
   printf("3) Scaling\n");
   printf("4) Shear X\n");
   printf("5) Shear Y\n");
   printf("6) Exit and print\n");
   scanf("%d",&choice);
   if (choice==1)
   {
     int ox,oy;
     printf("Enter new coordinates for Translation origin\n");
     scanf("%d %d",&ox,&oy);
     xa=xa+ox;
     xb=xb+ox;
     ya=ya+oy;
   yb=yb+oy;
 }
 else if(choice==2)
 {
   int rx,ry;
   double ang,val;
   printf("Enter coordinates for point about which should i rotate\n");
   scanf("%d %d",&rx,&ry);
   printf("Enter angle by which to rotate\n");
   scanf("%lf",&ang);
   xa=xa+rx;
   xb=xb+rx;
   ya=ya+ry;
   yb=yb+ry;

   val = PI / 180.0;
    ang=ang*val;
    float nxa=xa,nya=ya,nyb=yb,nxb=xb;

  xa = (nxa*cos(ang))-(nya*sin(ang));
  ya = (nxa*sin(ang))+(nya*cos(ang));
  xb = (nxb*cos(ang))-(nyb*sin(ang));
  yb = (nxb*sin(ang))+(nyb*cos(ang));
xa=xa-rx;
xb=xb-rx;
ya=ya-ry;
yb=yb-ry;

 }
 else if (choice==3)
 {
   int sx,sy;
   printf("Enter scaling factors for x and y directions\n");
   scanf("%d %d",&sx,&sy);
   int rx,ry;
   printf("Enter coordinates for point about which should i Scale\n");
   scanf("%d %d",&rx,&ry);
   xa=xa+rx;
   xb=xb+rx;
   ya=ya+ry;
   yb=yb+ry;

   xa=xa*sx;
   xb=xb*sx;
   ya=ya*sy;
```

```
     yb=yb*sy;

     xa=xa-rx;
     xb=xb-rx;
     ya=ya-ry;
     yb=yb-ry;
   }
   else if (choice==4)
   {
     int sx;
     printf("Enter Shear factors for x direction\n");
     scanf("%d",&sx);
     xa=xa+sx*ya;
     xb=xb+sx*yb;

   }
   else if(choice==5)
   {
     int sy;
     printf("Enter Shear factors for y direction\n");
     scanf("%d",&sy);
     ya=ya+sy*xa;
     yb=yb+sy*xb;
   }

}


  initgraph(&gd,&gm,NULL);
  line(xa,ya,xb,yb);
  setlinestyle(DASHED_LINE,0,THICK_WIDTH);
  line(xao,yao,xbo,ybo);
  delay(5000);
  closegraph();
  return 0;
}
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ gcc prac_7_2DComposite_transformation_algo.c -lgraph -lm
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the starting point
10
10
Enter the ending point
100
100
Enter the type of transformation from list given
1) Translation
2) Rotation
3) Scaling
4) Shear X
5) Shear Y
6) Exit and print
1
Enter new coordinates for Translation origin
60
70
Enter the type of transformation from list given
1) Translation
2) Rotation
3) Scaling
4) Shear X
5) Shear Y
6) Exit and print
2
Enter coordinates for point about which should i rotate
1
2
Enter angle by which to rotate
-10
Enter the type of transformation from list given
1) Translation
2) Rotation
3) Scaling
4) Shear X
5) Shear Y
6) Exit and print
6
[xcb] Unknown sequence number while processing queue
```



DASHED_LINE is the original line.

Practical 8

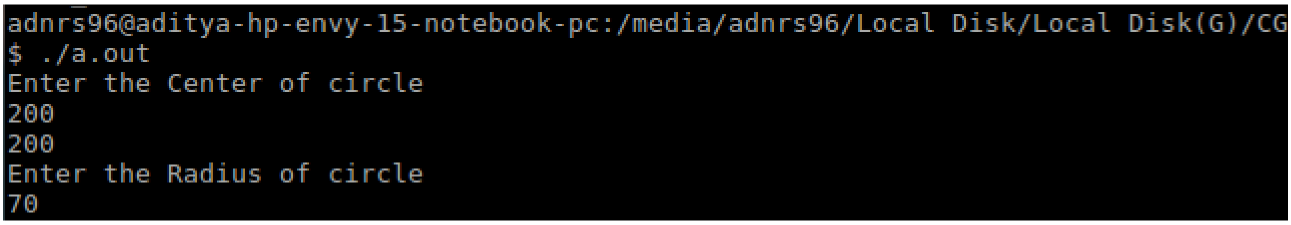
Task 1 : Write a program to demonstrate Bresenham’s mid point circle algorithm.

Source Code:

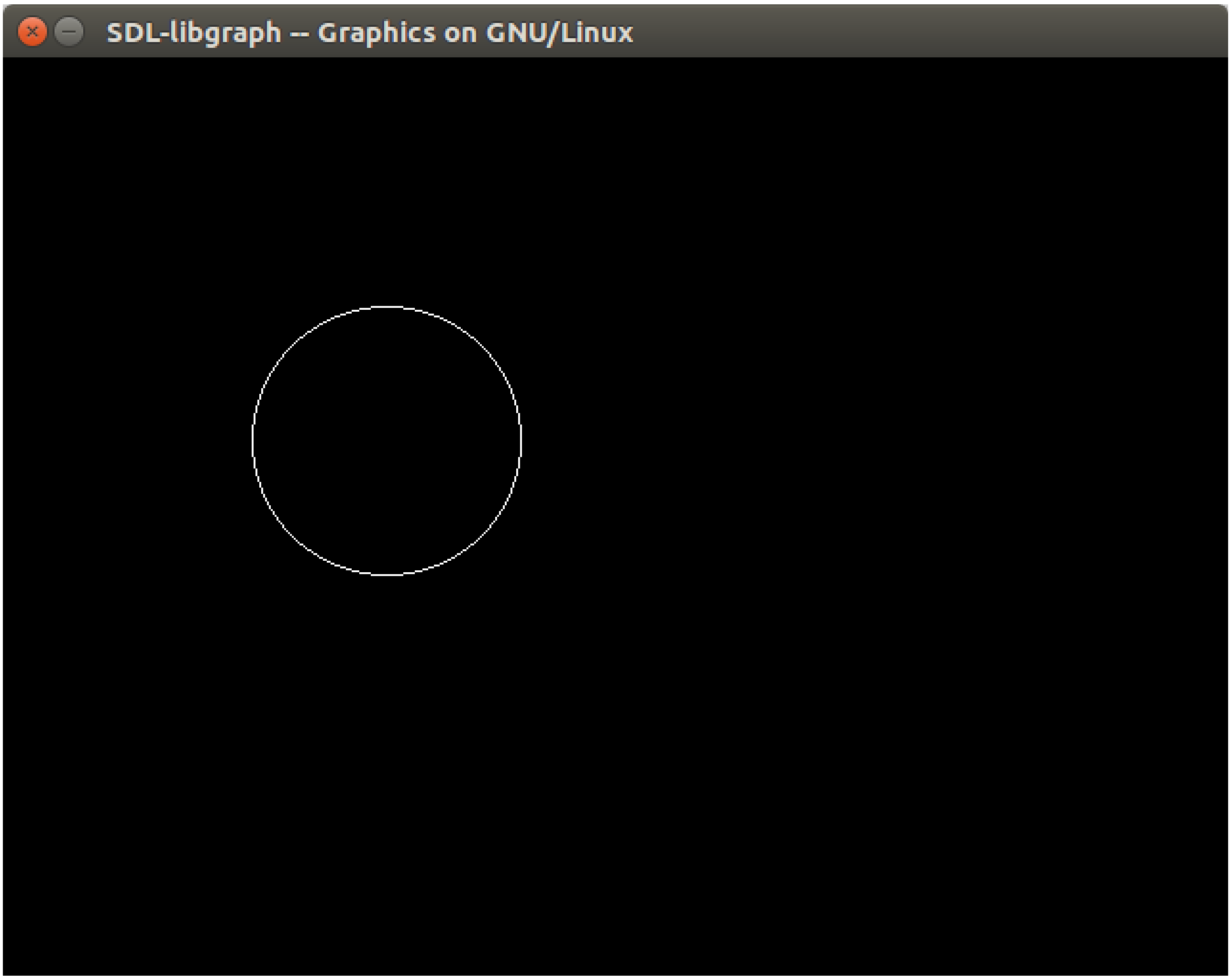
```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
int main()
{
   int gd = DETECT,gm;
   float xa,ya,r;
   printf("Enter the Center of circle\n");
   scanf("%f %f",&xa,&ya);
   printf("Enter the Radius of circle\n");
   scanf("%f",&r);
   initgraph(&gd,&gm,NULL);
   float pk = (5/4)-r;
   float x = 0,y=r;
   while(y>=x)
   {
     putpixel(x+xa,y+ya,WHITE);
     putpixel(y+xa,x+ya,WHITE);
     putpixel(x+xa,-(y)+ya,WHITE);
     putpixel(y+xa,-(x)+ya,WHITE);
     putpixel(-(x)+xa,y+ya,WHITE);
     putpixel(-(y)+xa,x+ya,WHITE);
     putpixel(-(x)+xa,-(y)+ya,WHITE);
     putpixel(-(y)+xa,-(x)+ya,WHITE);
     if(pk<0)
     {
       x+=1;
       pk=pk+2*x+1;
     }
     else
     {
       x+=1;
       y-=1;
       pk=pk+2*x+1-2*y;
     }
   }
```

```
    delay(5000);
    closegraph();
    return 0;
  }
```

Output:



```
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the Center of circle
200
200
Enter the Radius of circle
70
```



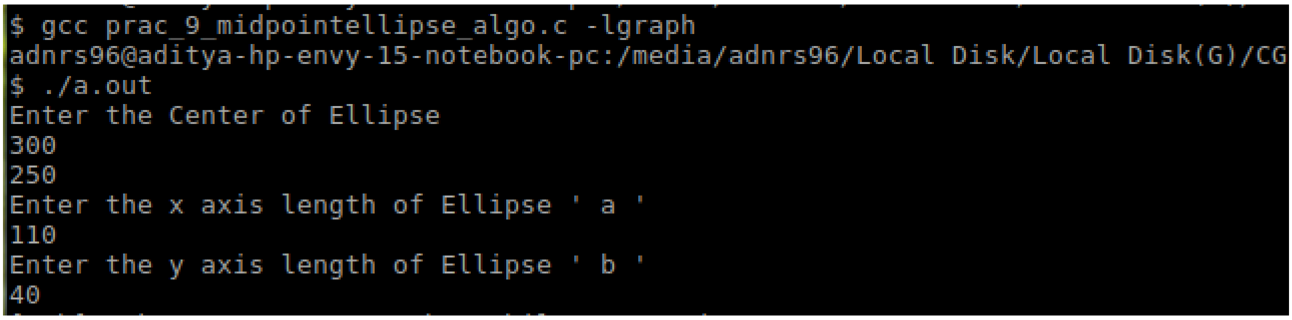
Practical 9

Task 1 : Write a program to demonstrate Bresenham’s mid point ellipse algorithm.

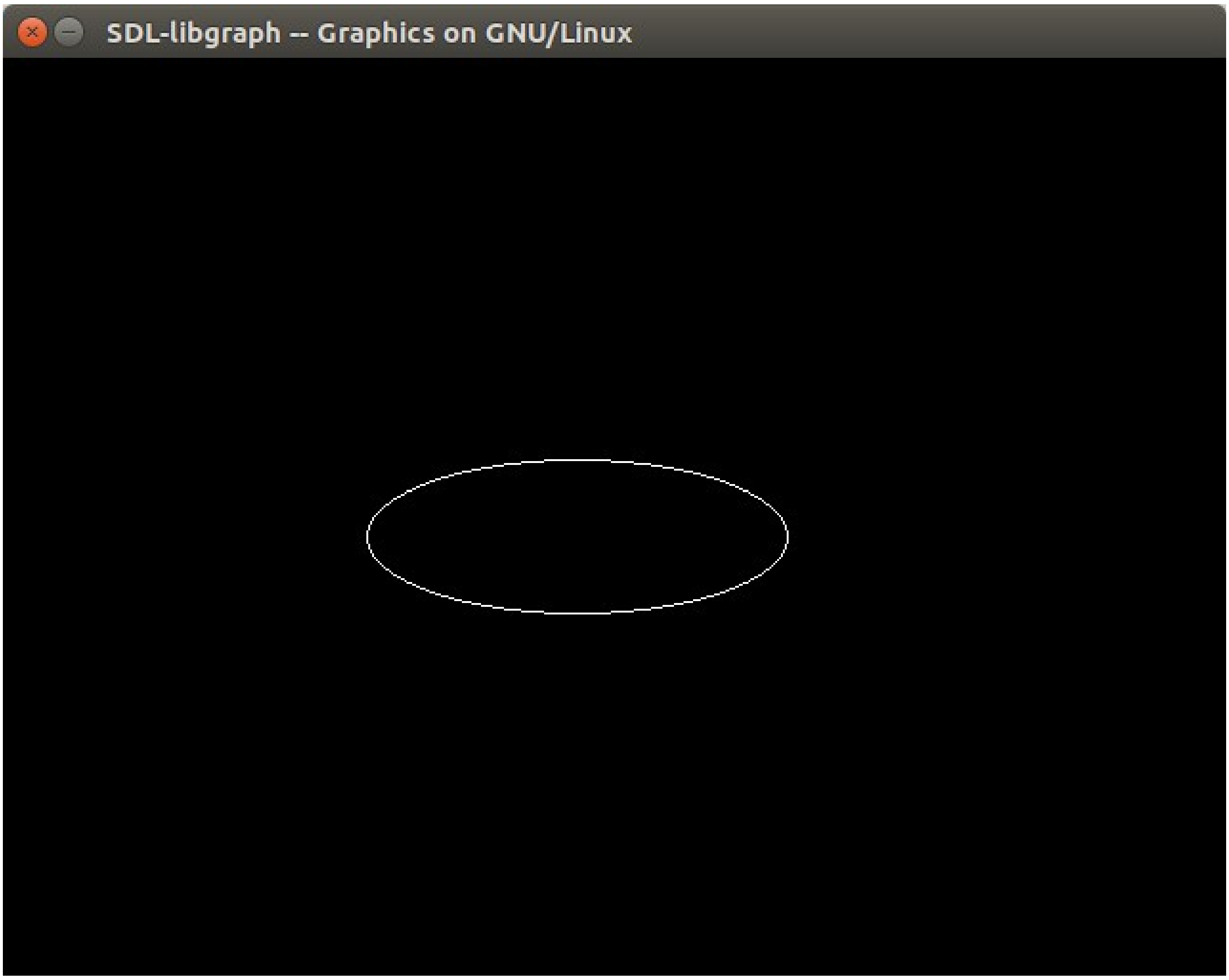
Source Code:

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
void ellipseDrawPoints(float x_center,float y_center,float x,float y)
{
  putpixel(x_center+x,y_center+y,WHITE);
  putpixel(x_center-x,y_center+y,WHITE);
  putpixel(x_center+x,y_center-y,WHITE);
  putpixel(x_center-x,y_center-y,WHITE);
}
int main()
{
   int gd = DETECT,gm;
   float xa,ya,ra,rb;

   printf("Enter the Center of Ellipse\n");
   scanf("%f %f",&xa,&ya);
   printf("Enter the x axis length of Ellipse ' a '\n");
   scanf("%f",&ra);
   printf("Enter the y axis length of Ellipse ' b '\n");
   scanf("%f",&rb);
   initgraph(&gd,&gm,NULL);
   float x = 0,y=rb;
   float px = 0,py = 2*ra*ra*y,p=rb*rb-(ra*ra*rb)+(0.25*ra*ra);
   ellipseDrawPoints(xa,ya,x,y);
   while(px<py)
   {
     x++;
     px+=2*rb*rb;
     if(p<0)
     {
       p+=rb*rb+px;
     }
     else
     {
        y--;
        py-=2*ra*ra;
        p+=rb*rb+px-py;
      }
     ellipseDrawPoints(xa,ya,x,y);
   }
  p=rb*rb*(x+0.5)*(x+0.5)+ra*ra*(y-1)*(y-1)-ra*ra*rb*rb;
  while(y>0)
  {
    y--;
    py-=2*ra*ra;
    if(p>0)
    {
      p+=ra*ra-py;
    }
    else
    {
       x++;
       px+=2*rb*rb;
       p+=ra*ra+px-py;
    }
    ellipseDrawPoints(xa,ya,x,y);
 }

   delay(5000);
   closegraph();
   return 0;
}
```

Output:



```
$ gcc prac_9_midpointellipse_algo.c -lgraph
adnrs96@aditya-hp-envy-15-notebook-pc:/media/adnrs96/Local Disk/Local Disk(G)/CG
$ ./a.out
Enter the Center of Ellipse
300
250
Enter the x axis length of Ellipse ' a '
110
Enter the y axis length of Ellipse ' b '
40
```


SDL-libgraph -- Graphics on GNU/Linux