Name- Manish Singh

Course- BCA 'C'

Roll No- 1121079

Q.

```
#include<stdio.h>
#include<graphics.h>
int main()
{
  int rou(float num)
  {
  return num<0?num-0.5:num+0.5;
  }
  int x1=100, x2=300, y1=100, y2=200;
  int gd=Detect, gm;
  float pk, pkk, x, y, step;
  int dx=x2-x1;
  int dy=y2-y1;
  Pk = 2*dx-dy;
  if(dx>dy)
   step=dx;
   else
   step=dy;
  initgraph(&gd,&gm,"");
  outtextxy(x1,y1,"A");
  outtextxy(x2,y2,"B");
  Putpixel(x1, y1, WHITE);
  x=x1, y=y1;
  While(step>0)
  {
   if(pk<0)
  {
   pkk=pk+2*dy;
  }
  else
  {
   pkk = pk+2*dy-2*dy;
```

```
Y++;
step--;
}
getch();
return();
}
```

Bresenham's Line Algorithm

Step1: Start Algorithm

Step2: Declare variable
$x_1, x_2, y_1, y_2, d, i_1, i_2, dx, dy$

Step3: Enter value of $x_1, y_2 x_2, y_2$
where x_1, y_1 are coordinates of
starting point
and x_2, y_2 are coordinates of Ending point

Step4: Calculate $dx = x_2 - x_1$
Calculate $dy = y_2 - y_1$
Calculate $i_1 = 2 * dy$
Calculate $i_2 = 2 * (dy - dx)$
Calculate $d = i_1 - dx$

Step5: Consider (x, y) as starting point and xend as maximum poible value of x.
then $x = y_2$
$Y = y_2$
$xend = x_1$
if $dx > 0$
then $x = x_1$
$Y = y_1$
$xend = x_2$

Step6: Generate point at (x, y) coordinates.

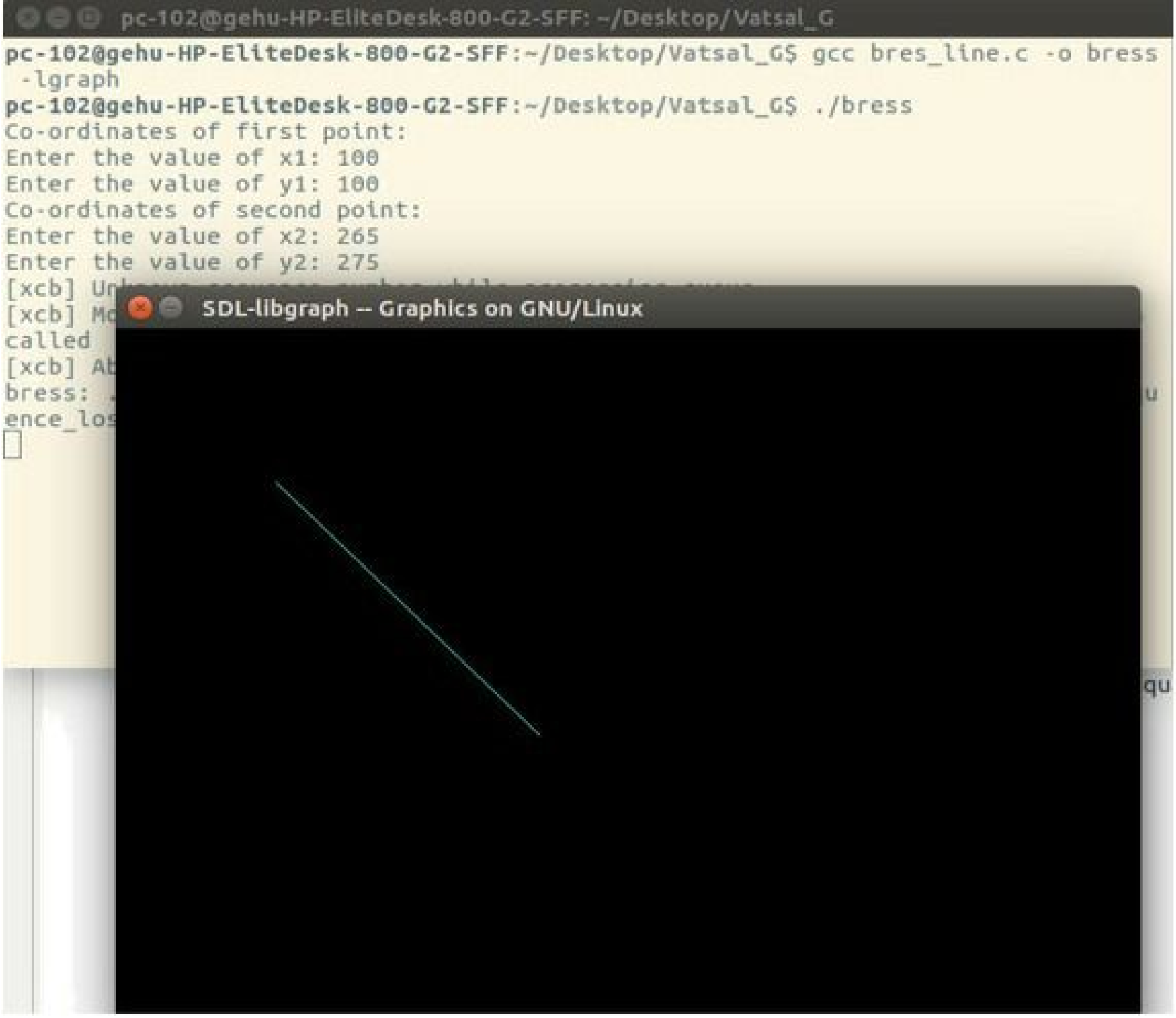
Step7: Check if whole line is generated.
if $x >= xend$
stop.

Step8: increment x = x+1

Step9: ~~increment x~~ Draw a point of latest (x,y) coordinates

Step11: Go to step 7

Step12: End of Algorithm


pc-102@gehu-HP-EliteDesk-800-G2-SFF: ~/Desktop/Vatsal_G
pc-102@gehu-HP-EliteDesk-800-G2-SFF:~/Desktop/Vatsal_G$ gcc bres_line.c -o bress
-lgraph
pc-102@gehu-HP-EliteDesk-800-G2-SFF:~/Desktop/Vatsal_G$ ./bress
Co-ordinates of first point:
Enter the value of x1: 100
Enter the value of y1: 100
Co-ordinates of second point:
Enter the value of x2: 265
Enter the value of y2: 275
SDL-libgraph -- Graphics on GNU/Linux

Name- Manish Singh

Course- BCA 'c'

Roll No- 1121079

Q2

```
#include <stdio.h>
#include<graphics.h>
Void drawcircle(int x0, int y0, int radius)
{
 int x = radius;
 int y = 0;
 int err=0;
while(x >=y)
{
 Putpixel(x0+x, y0+y, 7);
 Putpixel (x0+y, y0+x, 7);
 Putpixel(x0-y, y0+x, 7);
 Putpixel (x0-x, y0+y, 7);
 Putpixel(x0-x, y0-y, 7);
 Putpixel(x0-y, y0-x, 7);
 Putpixel(x0+y, y0-x, 7);
 Putpixel (x0+x, y0-y, 7);
 if (err<=0)
 {
    y+=1;
    err+=2*y+1;
 }
  if(err >0)
 {
    x-=1;
    err-=2*x+1
}
}}
```

```
int main()
{
 int gdriver = DETECT, gmode, error, x, y, r;
 Printf("Enter radius of circle"); scanf("%d", &r);
 Printf("Enter co-ordinates of center (x and y): ");
 Scanf("%d %d", &x, &y)
 initgraph(&gdriver, &gmode, "");
 drawcircle(x, y, r);
 delay(9999999);
 return 0;
```

Algorithm

Step1: Assign the starting point coordinates (x_0, y_0)

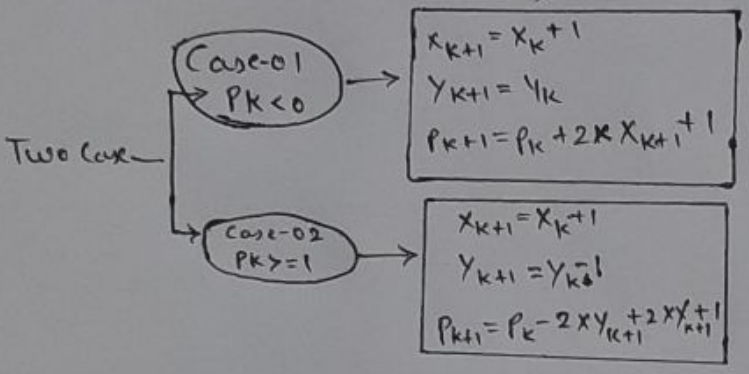
as- $x_0 = 0$

$y_0 = R$

Step2: Calculate the value of intial decision parameter P_0 as

$$P_0 = 1 - R$$

Step3: suppose the current point is (x_k, y_k) and the next point is (x_{k+1}, y_{k+1})

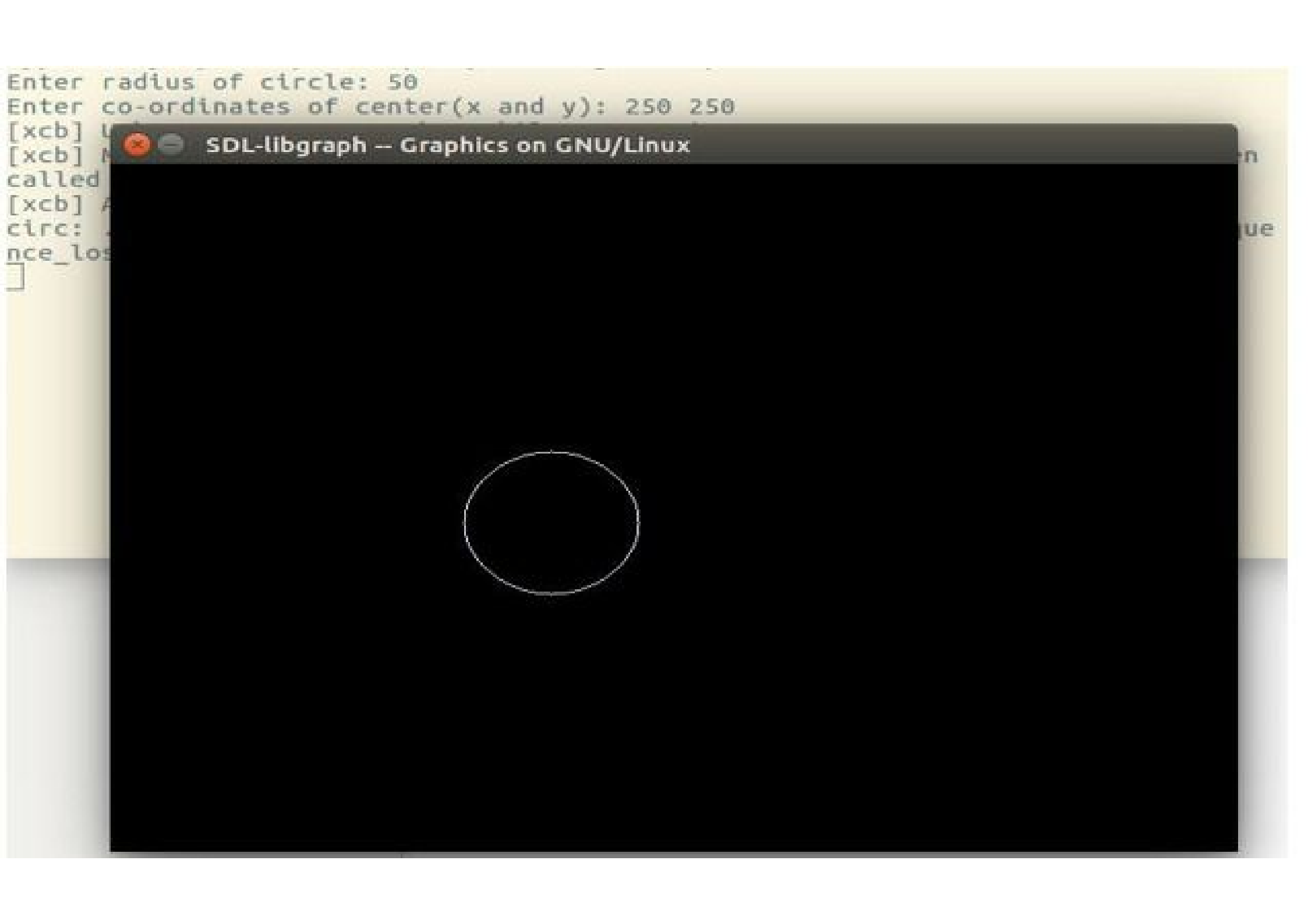


Step-04- centre point (x_0, y_0) is not $(0,0)$ then do the following and plot point

- $x_{plot} = x_c + x_0$
- $x_{plot} = y_c + y_0$

Step5- keep, (x_c, y_c) denotes the current value of x and y coordinates

Step6:- Step-05 generates all the points for one octant to find the points for other seven Octant,


Enter radius of circle: 50
Enter co-ordinates of center(x and y): 250 250
SDL-libgraph -- Graphics on GNU/Linux