Name - Vivek Thapa
University rollno. - 1121179
Subject - CG Practicle

Bresenham's Line Algorithm

Qs 1
Ans ->

```
#include <stdio.h>
#include<graphics.h>
 int main()
 {
  int gd = DETECT, gm, x0, y0, x1, y1, dx, dy, p, x, y;
  printf("Co-ordinates of first point:");
  Print f("\n Enter the value of x1:");
   scanf("%d", &x0);
  printf("Enter the value of y1:");
   scanf("%d", &y0);
  print f("Co-ordinates of second point:").
  print f("\nEnter the value of x2:").
   Scan f("%d", &x1).
   print f("Enter the value of y2:").
    Scan f("%d", &y1);
    initgraph(&gd, &gm, "");
    dx = x1 - x0;
    dy = y1 - y0;
    x = x0;
    y = y0;
    p = 2*dy-dx;
```

```
while (x<x1)
 {
 if (p>=0)
 {
 putpixel (x,y,4);
 y=y+1;
 p=p+2*dy-2*dx;
 }
 else
 {
 putpixel (x,y,4);
p=p+2*dy;
}
x=x+1;
}
getch();
return 0;
```

ALGORITHM :-

Step 1:- Start Algorithm

Step2:- Declare varible
x1,x2,y1,y2,d,i1,i2,dx,dy

Step3:- Enter value of x1,y1,x2,y2
where x1, y1 are coordinates of Ending point.
~~step 4~~ Ad x2,y2 are coordinats of Ending point

```
Step 4   Calculate dx = x2 - x1
         Calculate dy = y2 - y1
         Calculate i2 = 2*(dy - dx)
         Calculate d = i2 - dx
```

Step 5 Consider (x, y) as starting point and xend as maximum possible value of x.

```
    if dx < 0
    Then x = x2
         y = y2
         xend = x1
    if dx > 0
     Then x = x1
          y = y1
          xend = x2
```

Step 6 Generate point at (x, y) coordinates

Step 7 Check if whole line is generated

```
    if x >= xend
    Stop.
```

Step 8 Calculate co-ordinates of the next pixel

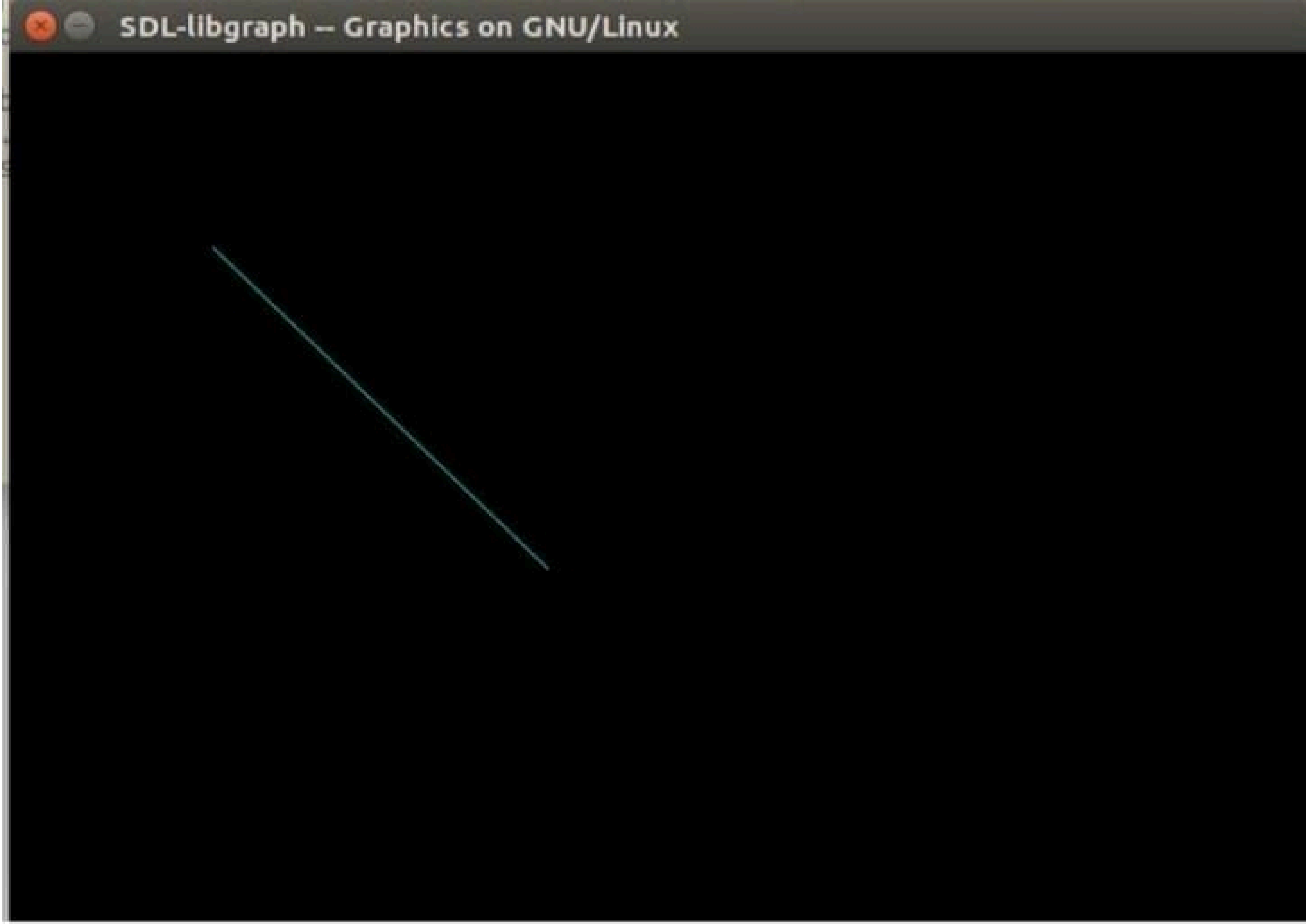
```
pixel if d < 0
        Then d = d + i2
          if d ≥ 0
          Then d = d + i1
          if d ≥ 0
        Then d = d + i2
             increment = y = y + 1
```

Step 9 Increment $x=x+1$

Step 10 Draw a point of latest (x,y] coordinates

Step 11 Go to step 7

Step 12 End of Algorithm



SDL-libgraph -- Graphics on GNU/Linux

Qs ②

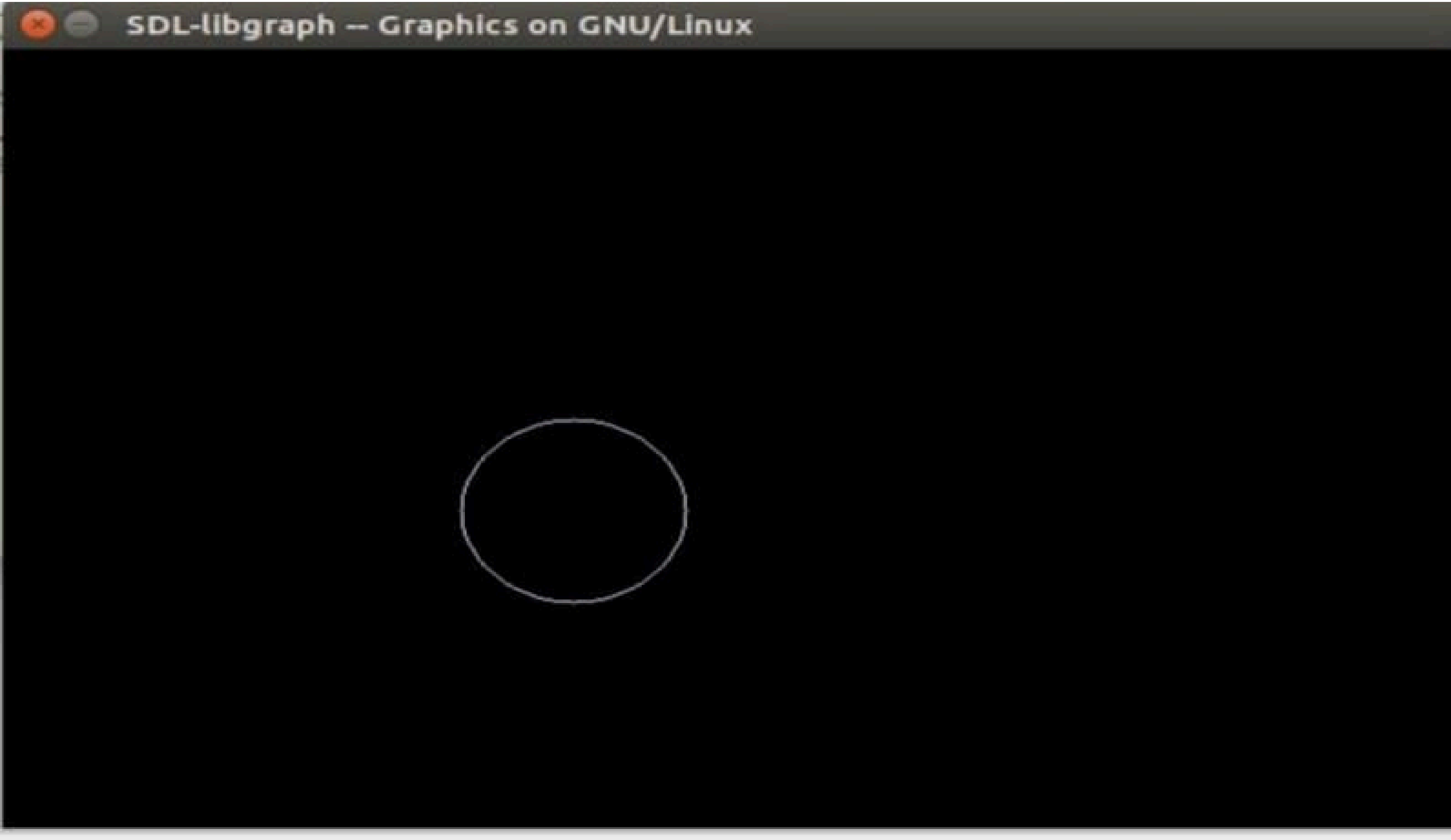
Ans

```
# include <Stdio.h>
# include <graphics.h>
void drawcircle (int x0, inty0 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);
putpivel (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 ()
{
intgddnive = DETECT, gmode ,error,x,y,r;
print f ("Enter radiow of circle :"); scanf("%d",
  &r);
printf ("Enter co-ordinates of center(x and y):");
Scan f ("%d %d", &x,&y);
intgraph (&gdriver, &gmode, "");
draw circle (x,y,r);
delay (9.99 9999);
  return 0;
```

(6)

Algorithm:-

```
Step(1) -> Start
Step(2) -> Put $x = 0, y = r$ in equation 2 we have $P = 1 - r$
 Step(3): Repeat steps while $x \le y$
       Plot(x,y)
       if (p < 0)
  Then set $p = p + 2x + 3$
     Elese
         $p = p + 2(x - y) + 5$
         $y = y - 1$ (end if)
         $x = x + 1$ (end loop)
Step(4): End
```


SDL-libgraph -- Graphics on GNU/Linux

calculate dy = y2 - y1

Qs 3

Ans

```
#include <graphics.h)
#include <stdio.h)
void boundary-fill (int x, int y, int fill-color,
int bound color)
{
if (get pixel (x, y)!= fill-color &&
getpixel (x,y)! = bound-color)
{
putpixel (x, y, fill-color).
delay (1);
boundary -fill (x+1, y, fill -color, bound-color);
boundary -fill (x , y-1, fill- color, bound -color).
boundar - fill (x-1, y, fill-color, bound-color);
boundary - fill (x y+1 , fill-color, bound-color);
boundary- fill ( x-1, y-1 ; fill- color, bound-color).
boundary - fill (x+1 , y-1, fill- color, bound-co
                lor;
boundary- fill (x-1, y+1 , fill _color, bond_ color),
boundary =fill (x+1;y+1 , fill- color,
boundary-color);
 }
 }
```

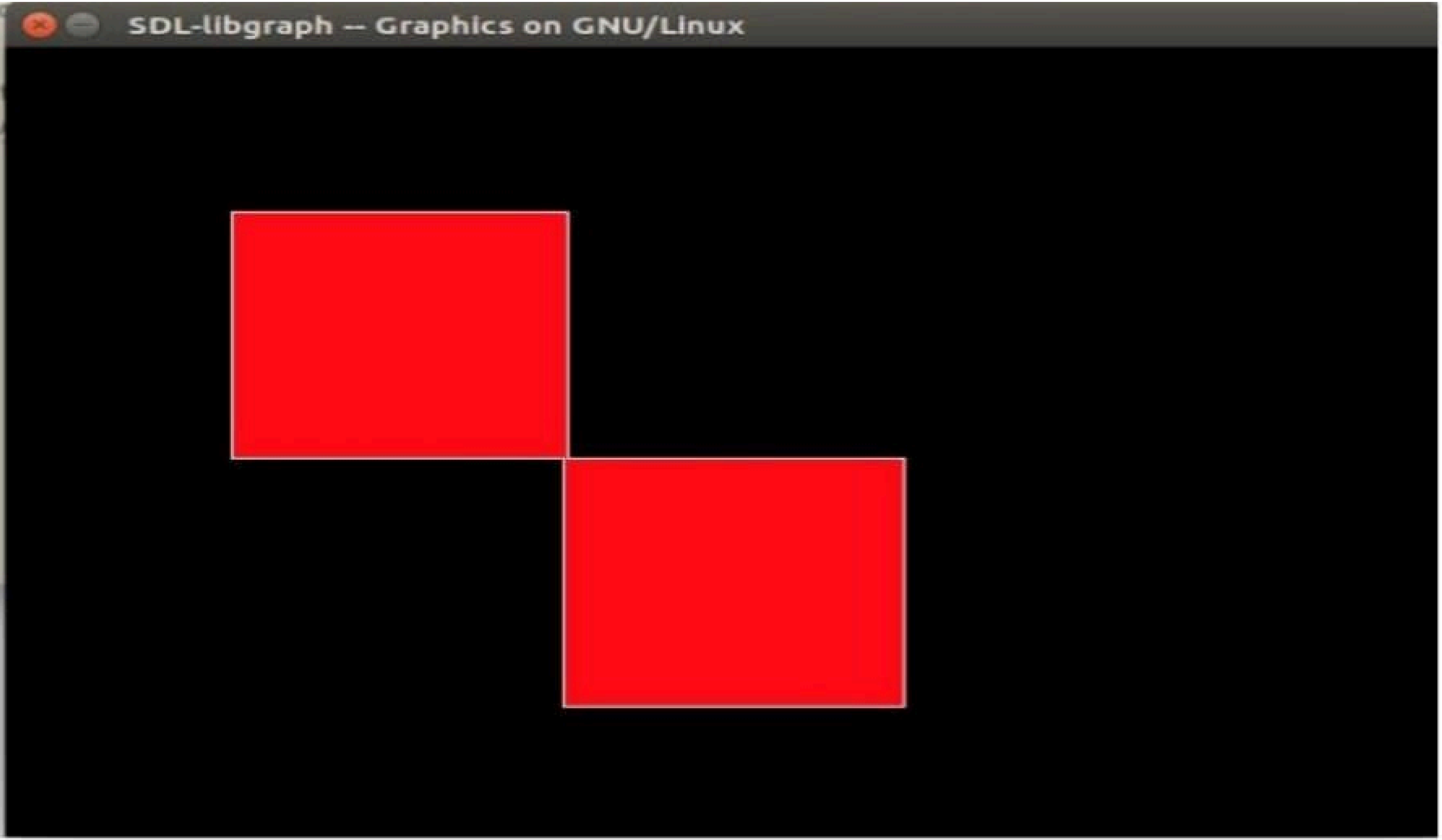
```
}
int main ()
{
int gd = DETECT, gm;
initgraph (&gd, &gm, "");
line (100,100,250,100);
line ((250,100,250,250);
line (250,250,400,250).
line (400,250,400,400);
line (248,400 ,400,400);
line (248,250 ,248,400);
line (100,100,100,250);
line (100,250,248,250);

  boundary_fild (150,150,RED,WHITE);
  getch();
  closegraph();
```

Algorithm:-

Start

1. Create a function named as boundary fil with 8 parameter.
2. Call it recursively until the boundary pixel are reached.
3. Stop



SDL-libgraph -- Graphics on GNU/Linux