Name :- Tanmoy Chauhan

Course:- BCA

Class Roll No:- 39

University Roll No:- 1121154

Subject:- Computer Graphics

1. Write an algorithm and pogram to inplement Bresham Line drawing algorithm.

A.

```
# 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.");
 printf("\n Enter the value of x1:");
 sconf("%d", & x0);
 printf(" Enter the value of y1:");
sconf("%d", & y0);
printf("co-ordinates of second point ");
Print f("\n Enter the value of y2:");
Scon f("%d", & y1);
intgraph(&gd, &gm, "");
dx =x1-x0;
dy= y1-y0;
x = x0;
y = y0;
p=2*dy-dx;
while (x<x1)
 {
 in(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 variable $x_1, x_2, y_1, y_2, d, i_1, i_2, dx, dy$

Step3:- Enter the value of x_1, y_1, x_2, y_2

where x_1, y_1 are coordinates of starting point.

And x_2, y_2 are coordinates of Ending point.

Step 4: calculate $dx = x_2 - x_1$

calcuate $dy = y_2 - y_1$

calculate $i_1 = 2 * dy$

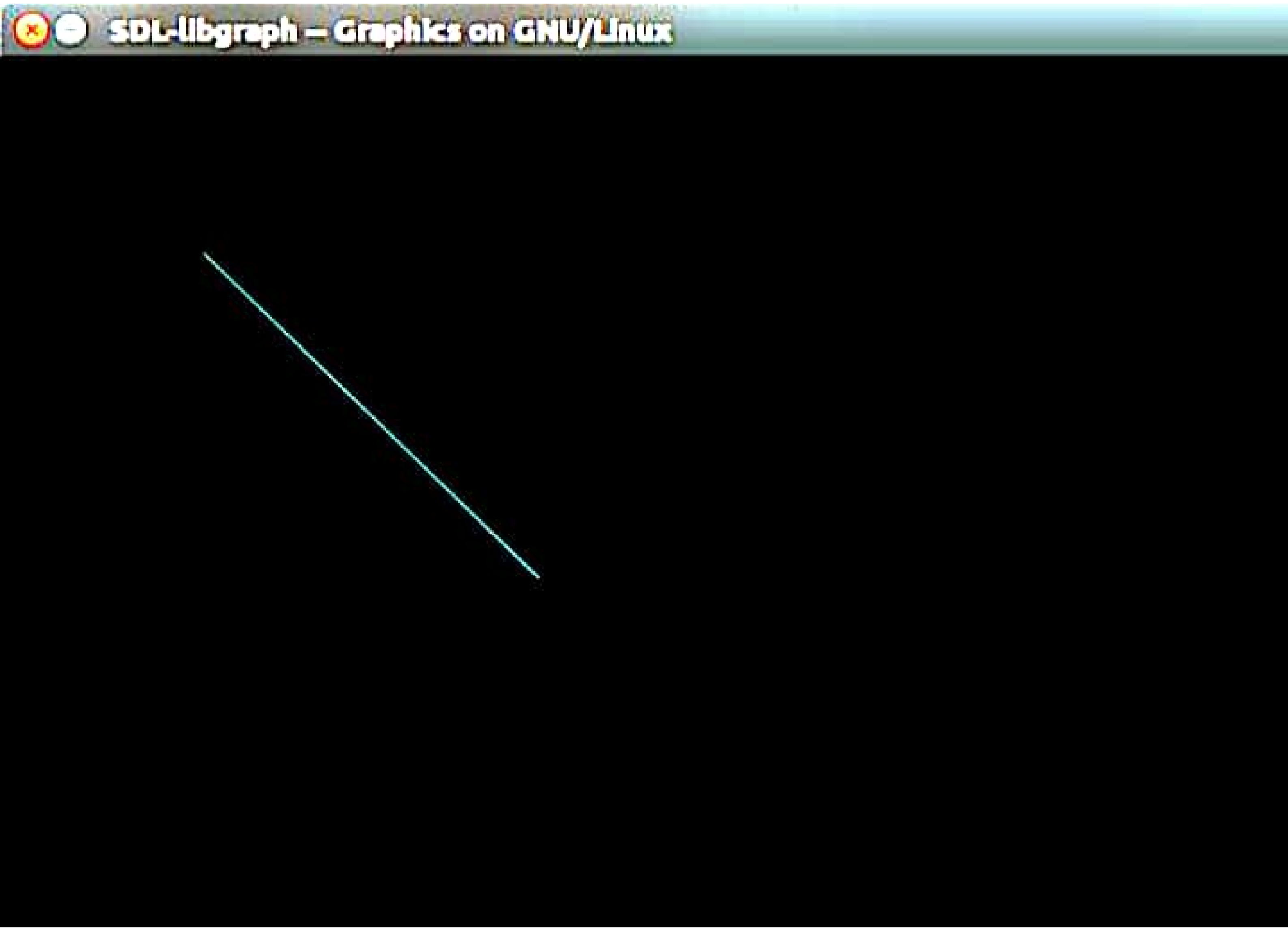
calculate $i_1 = 2 * (dy - dx)$

calculate $d = i_1 - dx$

Step5:- Calculate (x, y) as starting point and x end as maximum possible value of x.

```
    if $dx < 0$
    Then $x = x_2$
        $y = y_2$
        $x$ end $= x_1$
  if $dx > 0$
    Then $x = x_1$
    $y = y_1$
    $x$ end $= x_2$
```

```
Step 6:- Generate point at (x, y) coordinates
Step 7:- Check if whole line is generated
            if $x >= x_{end}$
            Stop.
Step 8:- Calculate co-ordinates of the next pixel
            if $d < 0$
            Then $d = d + i_1$
         if $d \geq 0$
         Then $d = d + i_2$
         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.
```

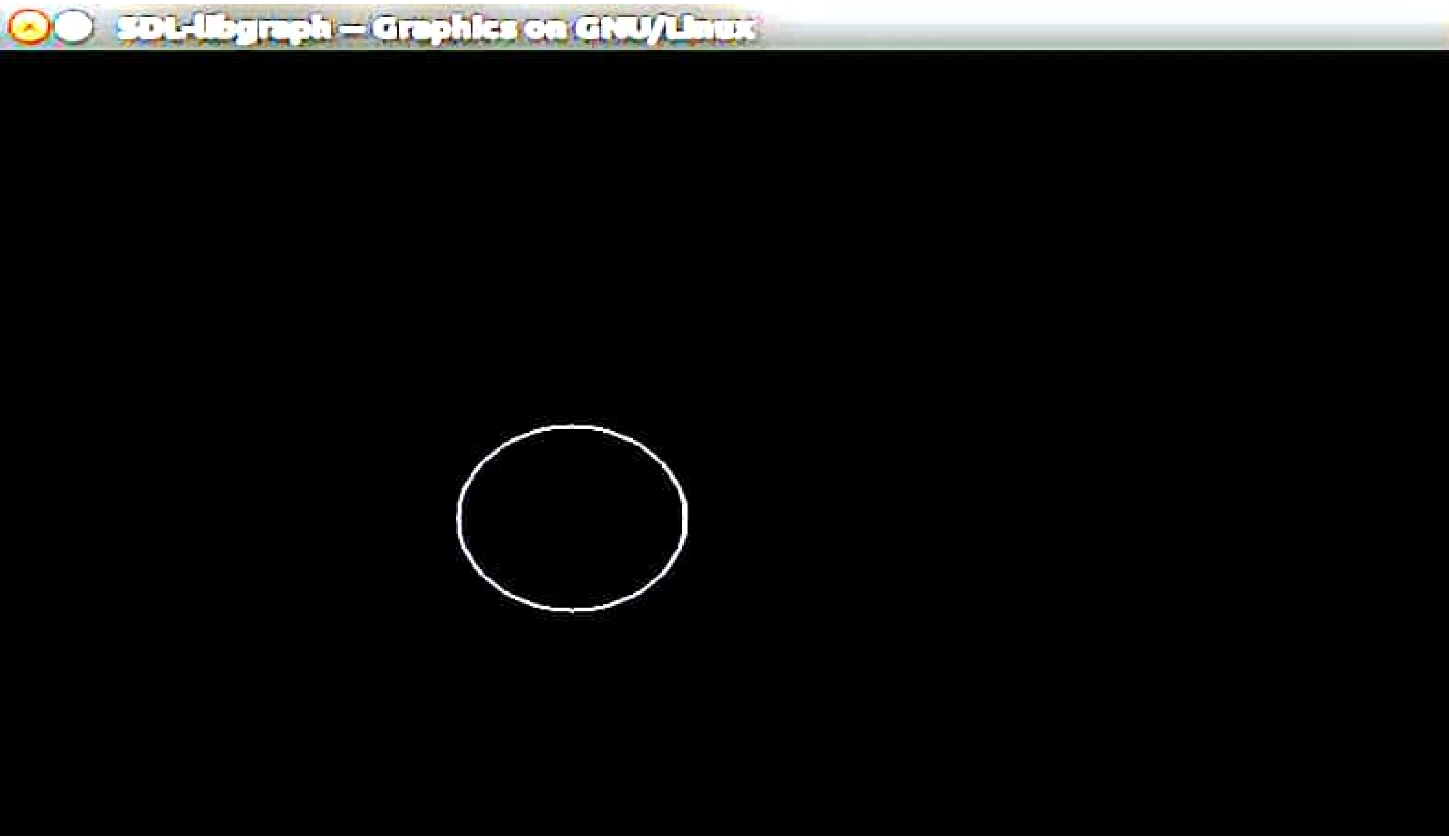

SDL-libgraph – Graphics on GNU/Linux

2. Write an algorithm and progrom to implement mid point circle drawing Algorithm.

```
# include<stdio.h>
# include < graphic.h>
void drawcircle ( int x0, int y0,int radius)
{

    int x= radius;
    int y = 0;
    int err=0;
   while ( x>=y )
   {
    putpipexl (x0+ y0+y,7);
    putpipexel (x0 +y, y0+x, 7);
    putpixel (x0-y, y0+x, 7);
    putpixll (x0-x, y0-y,7);
    putpixel (x0-y, y0-x, 7);
    putpixel ( x0-y, y0,-x ,7);
    putpixel (x0 +y, y0-x, 7);
    putpixll ( 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 co-ordinates of center(x and y): ");
     scanf ("%d" ~~&x)~~, %d", &x, &y);
     printf ("Enter ~~coordi~~ radius of circle : ");
     scanf ( "%d", &r);

     ~~initg~~
     initgraph (&gdriver, &gmode, "");
     drawcircle (x, y, r);
     delay ( 99999 );
       return 0;
     }
```

Algorithm :-

```
Step 1:-  Start
Step 2:-  Put $x = 0$, $y = r$ in equation 2
          we have $p = 1 - r$
Step 3:-  Repeat steps while $x \leq y$
            plot $(x, y)$
            if $(p < 0)$
    Then set $p = p + 2x + 3$
    Else
        $p = p + 2(x - y) + 5$
        $y = y - 1$ (end if)
        $x = x + 1$ (end loop)

  s ~~tep3 : Er~~
Step 4 : End.
```



SDL-libgraph — Graphics on GNU/Linux

3. Write an algorithm and program to implement boundary fill gtos algorithm using 8 connected approach.

```
# include < graphics.h >
# include< stdio.h >
void boundary_fill ( int x, int y, int fill_color, int bound_
                                          color )

  {
   if (getpixel (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);
   boundary_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- color);
    boundary-fill ( x - 1, y+1, fill -color, bound- color);
    boundary - fill (x+1, y+1, fill- color, bound- color);
    }
  }
  int main( )
    {
    int gd= DETECT, gm;
    initgraph (&gd, &gm);
    line (100, 100, 250, 100);
    line ( 250, 100, 250, 200);
    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-fill ( 150, 150, RED, WHITE);
   getch ();
  closegraph ();
}
```

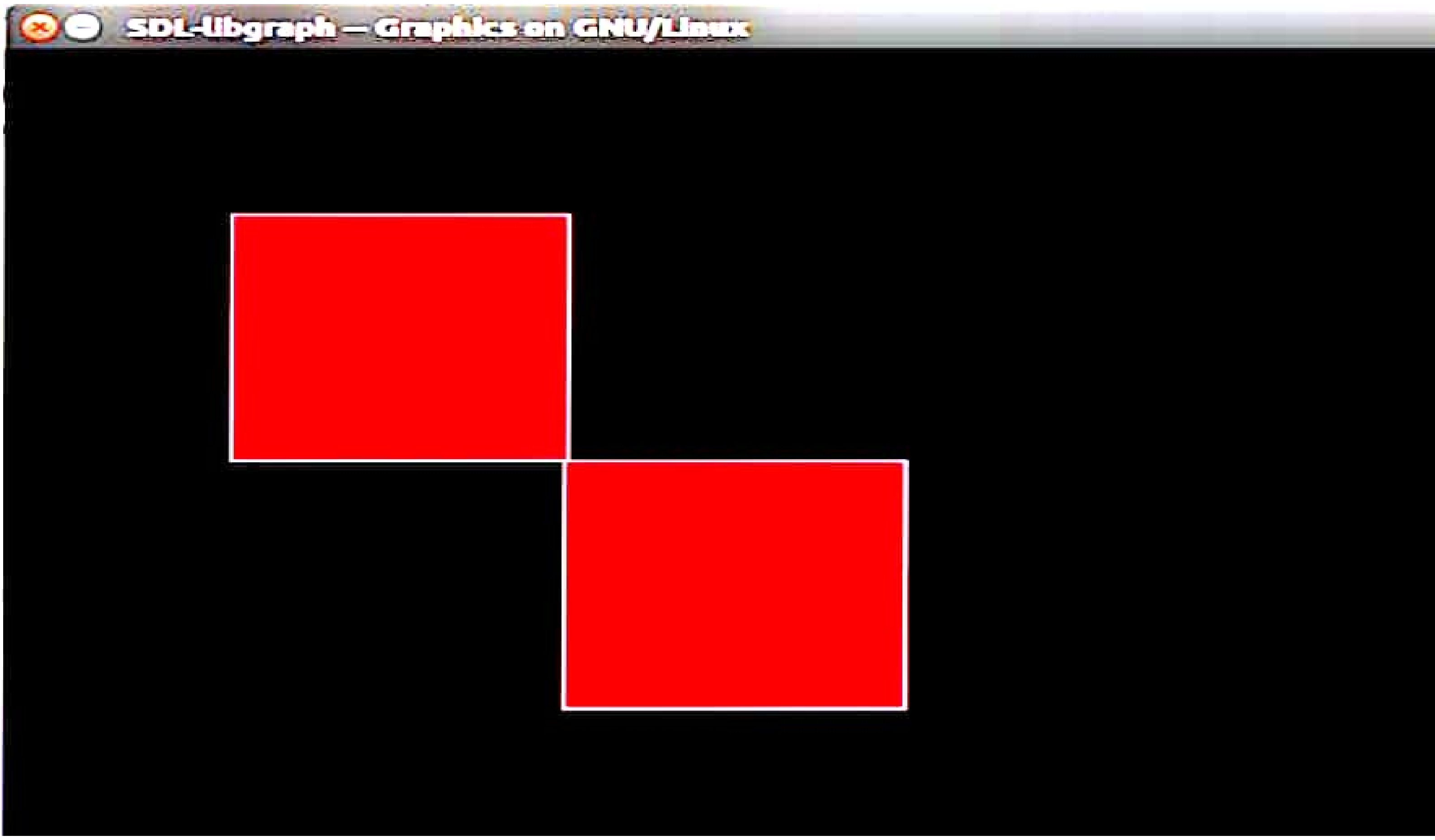
Algorithm.

Step 1:- Start

Step 2:- create a function named as boundary fill with 4 parameters. (x, y, f-color, b-color).

Step 3:- Call it recursively until the boundary pixels are reached.

Step 4:- Stop.



SDL-libgraph — Graphics on GNU/Linux