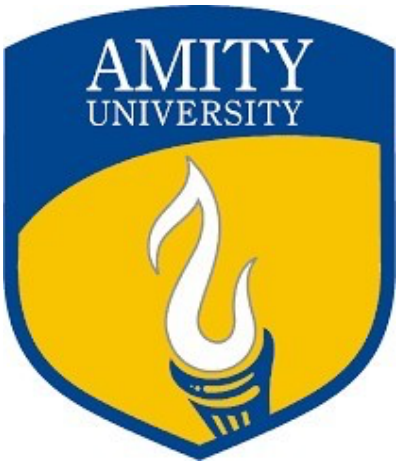
AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

AMITY UNIVERSITY, MANESAR GURUGRAM



Computer Graphics Lab File

SUBMITTED TO:

Dr.Shweta Sinha

SUBMITTED BY:

Aayush Verma
A50105220004
BTECH CSE A, 7th SEM (20-24)

Index

S. No	Topic	Date	Remarks	Teacher's Signature
1.	Write a program to change the working mode from text to graphics and plot a pixel.	14/08/2023		
2.	Write a program to display line, rectangle, circle, and polyline using graphics command.			
3.	Write a program to draw a smiley and a hut.	21/08/2023		
4.	Write a program to draw a pie chart and bar graph.	28/08/2023		
5.	Write a program to draw a line of slope between 0 and 1 using DDA Algorithm.	04/09/2023		
6.	Write a program to draw a line of slope between 0 and 1 using Bresenham's Line Drawing Algorithm.	11/09/2023		
7.	Write a program to draw a circle using Midpoint Circle Drawing Algorithm.	18/09/2023		
8.	Write a program to draw a circle using Bresenham's Circle Drawing Algorithm.	25/09/2023		

9.	Write a program to translate an object and plot it at a new position.	09/10/2023		
10.	Write a program to rotate a point (100, 50) about origin in clockwise direction.	16/10/2023		
11.	Write a program to scale an object to double of its size	30/10/2023		
12.	Write a program to fill a polygon using boundary-fill method.	06/11/2023		
13.	Write a program to polygon using flood-fill method.	20/11/2023		

Practical-1

Aim: Write a program to change the working mode from text to graphics and plot a pixel.

Source Code

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>


int main()

{

   // Initialize graphics driver and mode variables

   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");


   int x = 100;                    // Set x-coordinate

   int y = 200;                    // Set y-coordinate

   int color = RED;                // Set color


   // Put a pixel at the specified coordinates

   putpixel(x, y, color);


   getch();

   closegraph();
```

```
    return 0;
}
```

Output

Practical-2

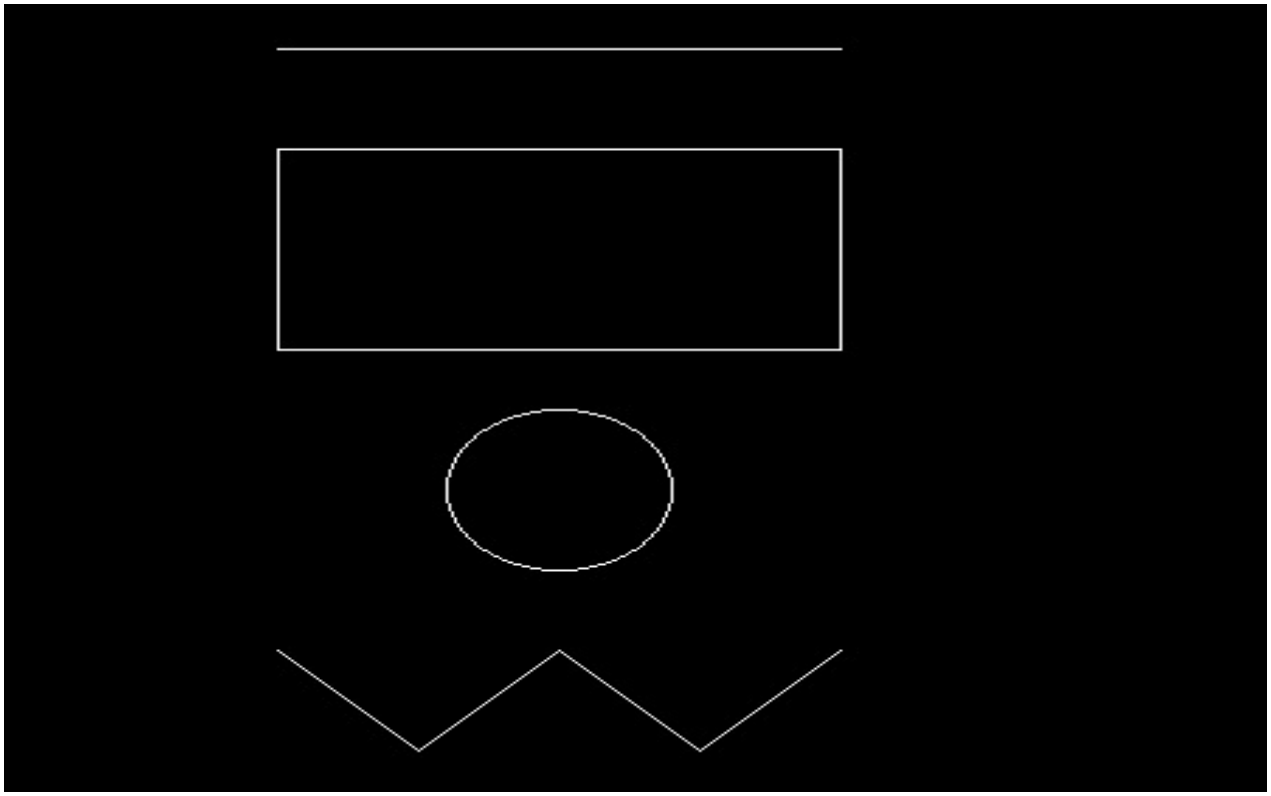
Aim: Write a program to display line, rectangle, circle, and polyline using graphics command.

Source Code

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>


int main()

{

   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");


   // Draw a line

   line(100, 100, 300, 100);


   // Draw a rectangle

   rectangle(100, 150, 300, 250);


   // Draw a circle

   circle(200, 320, 40);


   // Draw a polyline
```

```
    int polyPoints[] = {100, 400, 150, 450, 200, 400, 250, 450, 300, 400};
    drawpoly(5, polyPoints);
    getch();
    closegraph();

    return 0;
}
```

Output



Practical-3

Aim: Write a program to draw a smiley and a hut.

Source Code

Smiley

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

int main()
{
        int gr = DETECT, gm;
        initgraph(&gr, &gm, "C:\\Turboc3\\BGI");

        setcolor(YELLOW);

        circle(300, 100, 40);
        setfillstyle(SOLID_FILL, YELLOW);        // Set the fill style to solid fill
        floodfill(300, 100, YELLOW);                    // Flood fill the circle

        setcolor(BLACK);                        // Set the drawing color to black
        setfillstyle(SOLID_FILL, BLACK);
        fillellipse(310, 85, 2, 6);                     // Draw and fill an ellipse
```

```
        fillellipse(290, 85, 2, 6);
        // Draw an ellipse with specified parameters
        ellipse(300, 100, 205, 335, 20, 9);
        ellipse(300, 100, 205, 335, 20, 10);
        ellipse(300, 100, 205, 335, 20, 11);

        getch();
        closegraph();
        return 0;
}
```

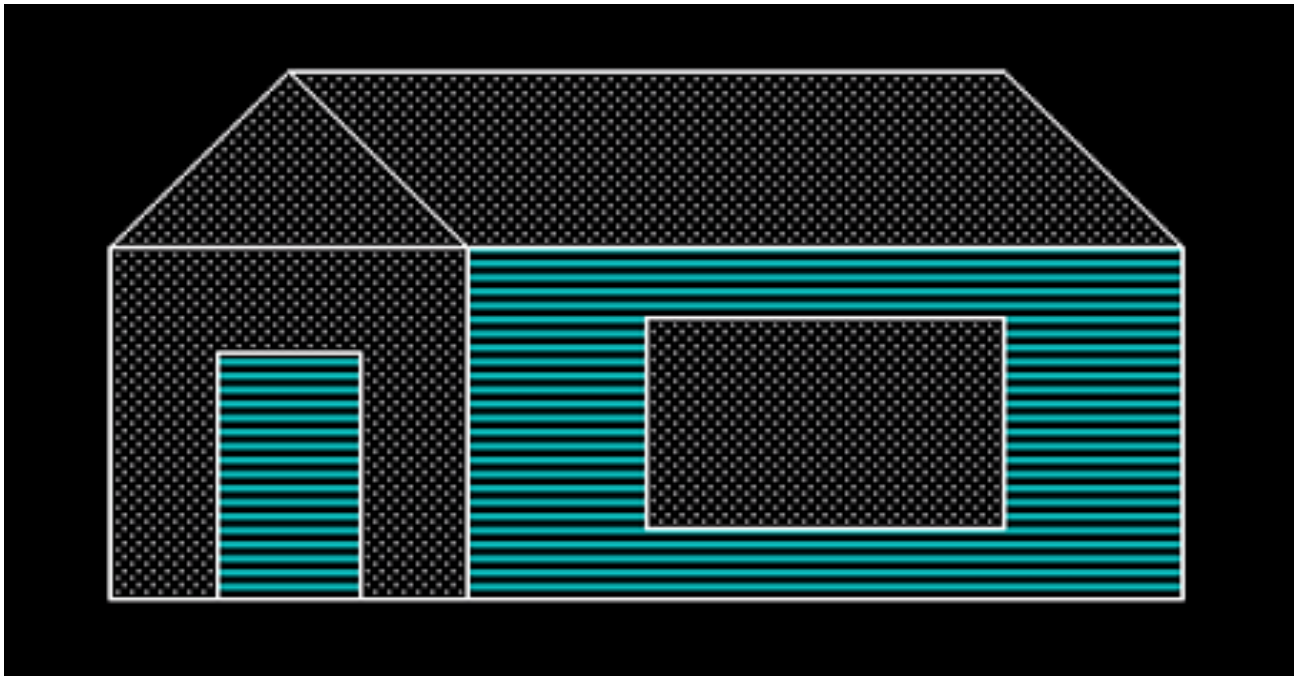
Output

Hut

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

void main()
{
        int gdriver = DETECT, gmode;
        initgraph(&gdriver, &gmode, "C:\\Turboc3\\BGI");

        line(100, 100, 150, 50);
        line(150, 50, 200, 100);
        line(150, 50, 350, 50);
        line(350, 50, 400, 100);

        rectangle(100, 100, 200, 200);
        rectangle(200, 100, 400, 200);
        rectangle(130, 130, 170, 200);
        rectangle(250, 120, 350, 180);
        setfillstyle(2, 3);

        floodfill(131, 131, WHITE);
        floodfill(201, 101, WHITE);
```

```
        setfillstyle(11, 7);
        floodfill(101, 101, WHITE);
        floodfill(150, 52, WHITE);
        floodfill(163, 55, WHITE);
        floodfill(251, 121, WHITE);

        getch();
        closegraph();
        return 0;
}
```

Output



Practical-4

Aim: Write a program to draw a pie chart and bar graph.

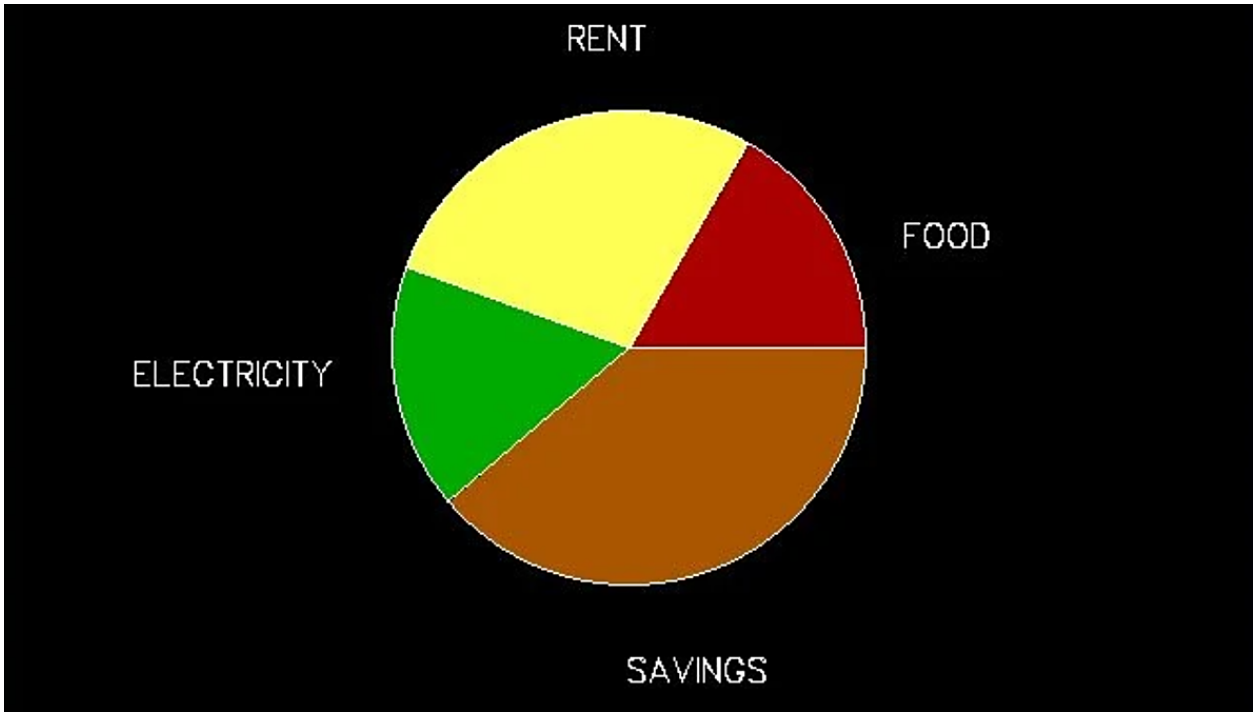
Source Code

Pie Chart

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

int main()
{
   int gd = DETECT, gm, x, y;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
   x = getmaxx() / 2;
   y = getmaxy() / 2;

   // Set the text style to sans-serif, horizontal direction, size 1
   settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 1);

   setfillstyle(SOLID_FILL, RED);
   pieslice(x, y, 0, 60, 120);
   outtextxy(x + 140, y - 70, "FOOD");
   setfillstyle(SOLID_FILL, YELLOW);
   pieslice(x, y, 60, 160, 120);
```

```
    outtextxy(x - 30, y - 170, "RENT");
    setfillstyle(SOLID_FILL, GREEN);
    pieslice(x, y, 160, 220, 120);
    outtextxy(x - 250, y, "ELECTRICITY");
    setfillstyle(SOLID_FILL, BROWN);
    pieslice(x, y, 220, 360, 120);
    outtextxy(x, y + 150, "SAVINGS");
    getch();
    closegraph();
    return 0;
}
```

Output

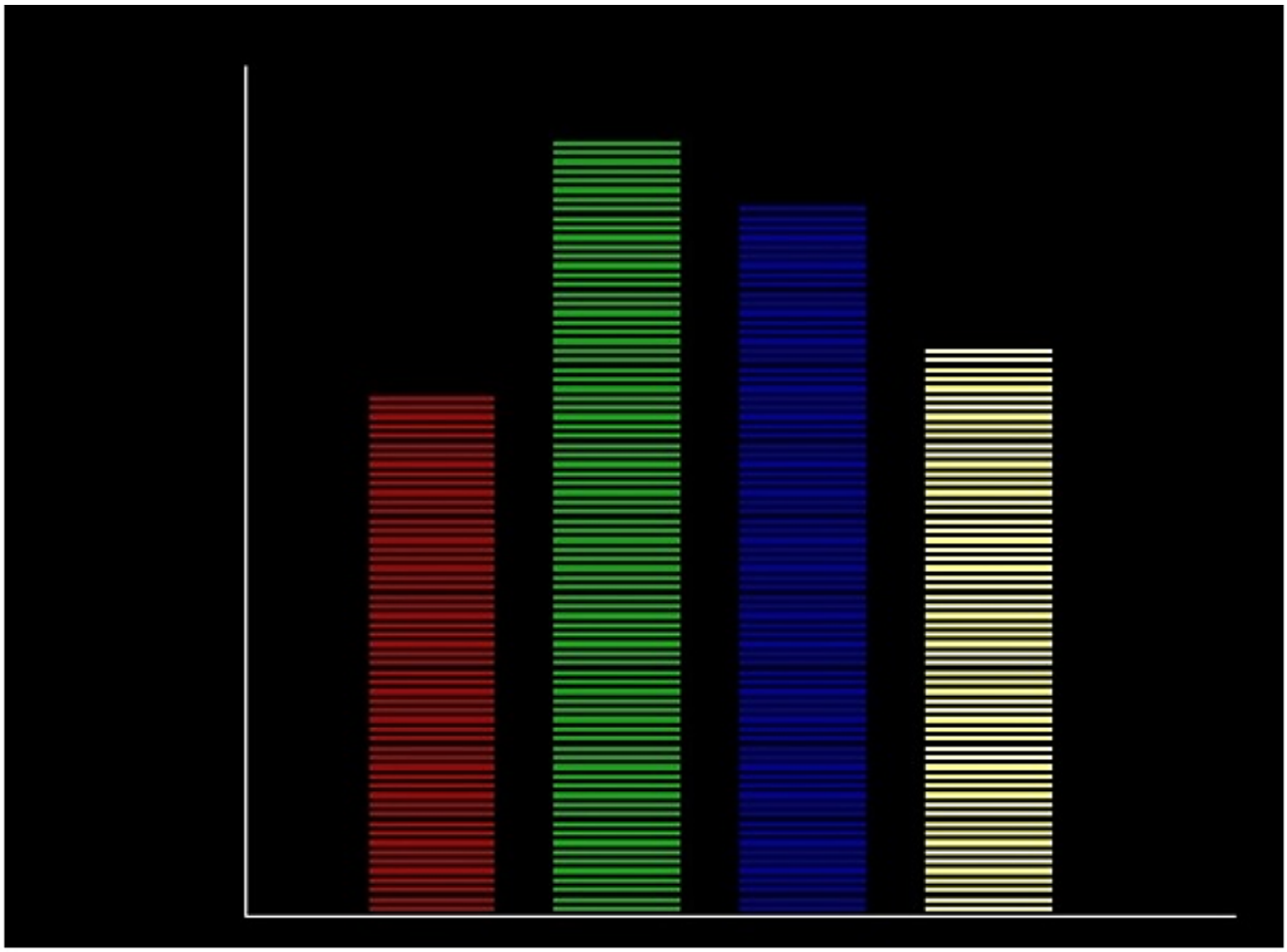


Bar Graph

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

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

   // Draw a vertical line from (100, 420) to (100, 60)
   line(100, 420, 100, 60);
   line(100, 420, 500, 420);

  // Set the fill style to a pattern (LINE_FILL)
   setfillstyle(LINE_FILL, RED);

   // Draw a vertical bar from (150, 200) to (200, 419)
   bar(150, 200, 200, 419);

   setfillstyle(LINE_FILL, GREEN);
   bar(225, 90, 275, 419);

   setfillstyle(LINE_FILL, BLUE);
```

```
    bar(300, 120, 350, 419);
    setfillstyle(LINE_FILL, YELLOW);
    bar(375, 180, 425, 419);

    getch();
    closegraph();

    return 0;
}
```

Output



Practical-5

Aim: Write a program to draw a line of slope between 0 and 1 using DDA Algorithm.

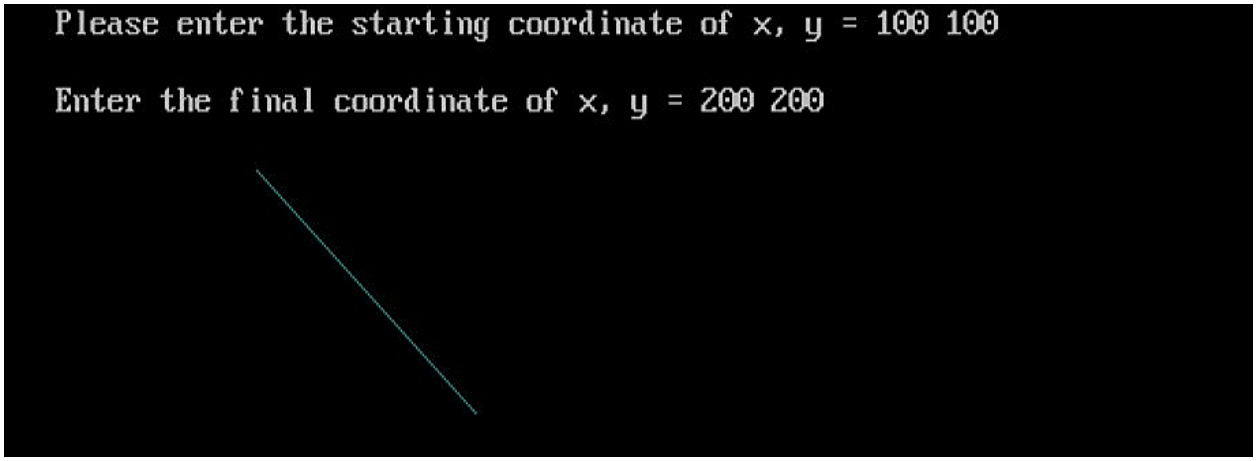
Source Code

```
#include <graphics.h>

#include <math.h>

#include <conio.h>


void main() {

   int x0, y0, x1, y1, i = 0;

   float delx, dely, len, x, y;


   int gr = DETECT, gm;

   initgraph(&gr, &gm, "C:\\Turboc3\\BGI");


   printf("Please enter the starting coordinate of x, y = ");

   scanf("%d %d", &x0, &y0);                    // Input starting coordinates


   printf("Enter the final coordinate of x, y = ");

   scanf("%d %d", &x1, &y1);                    // Input ending coordinates


   dely = abs(y1 - y0);                  // Calculate absolute y-difference

   delx = abs(x1 - x0);                  // Calculate absolute x-difference
```

```
 // Choose the longer dimension as the length
 if (delx < dely) {
      len = dely;
   }
else {
      len = delx;
   }

   delx = (x1 - x0) / len;                    // Calculate x-increment per step
   dely = (y1 - y0) / len;                    // Calculate y-increment per step

   x = x0 + 0.5;                  // Initialize x with a small offset for accuracy
   y = y0 + 0.5;                  // Initialize y with a small offset for accuracy

   do {
      putpixel(x, y, 3);
      x = x + delx;
      y = y + dely;
      i++;
      delay(30);
   }

while (i <= len);
```

```
    getch();
    closegraph();


    return 0;
}
```

Output



Practical-6

Aim: Write a program to draw a line of slope between 0 and 1 using Bresenham's Line Drawing Algorithm.

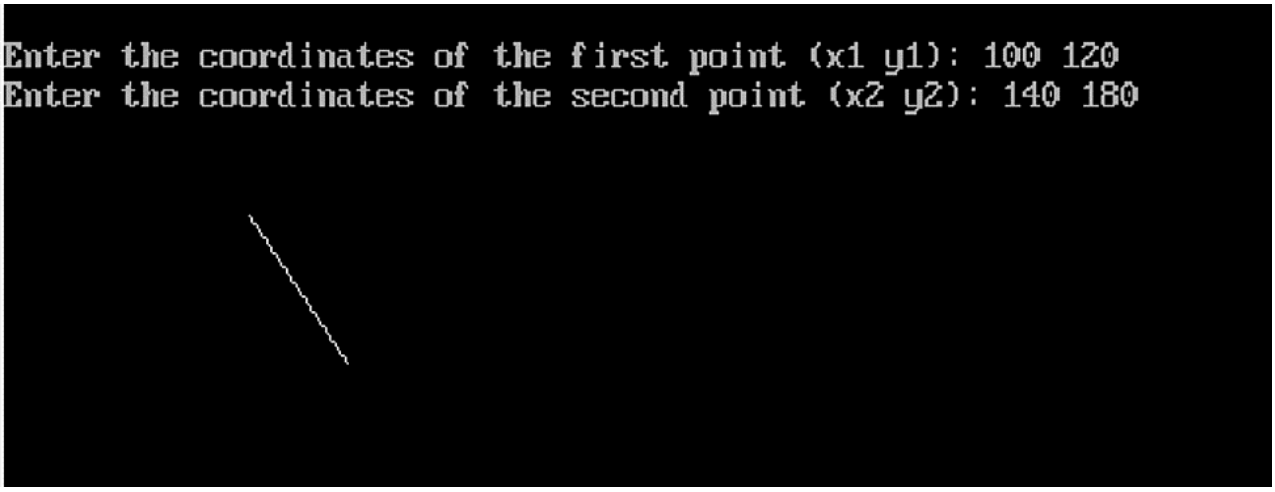
Source Code

```
#include <graphics.h>

#include <stdio.h>

#include <conio.h>


int main() {

   int x1, y1, x2, y2, dx, dy, sx, sy, err, e2;


   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");


   printf("Enter the coordinates of the first point (x1 y1): ");

   scanf("%d %d", &x1, &y1);


   printf("Enter the coordinates of the second point (x2 y2): ");

   scanf("%d %d", &x2, &y2);


   dx = abs(x2 - x1);

   dy = abs(y2 - y1);
```

```
if (x1 < x2)
   sx = 1;
else
   sx = -1;

if (y1 < y2)
   sy = 1;
else
   sy = -1;

err = dx - dy;                                  // Initialize error term

while (1) {
   putpixel(x1, y1, WHITE);

   if (x1 == x2 && y1 == y2) {
      break;
   }

   e2 = 2 * err;                                // Calculate double the error term

   if (e2 > -dy) {
      err = err - dy;
```

```
            x1 = x1 + sx;
        }

        if (e2 < dx) {
            err = err + dx;
            y1 = y1 + sy;
        }
    }

    getch();
    closegraph();

    return 0;
}
```

Output



Practical-7

Aim: Write a program to draw a circle using Midpoint Circle Drawing Algorithm.

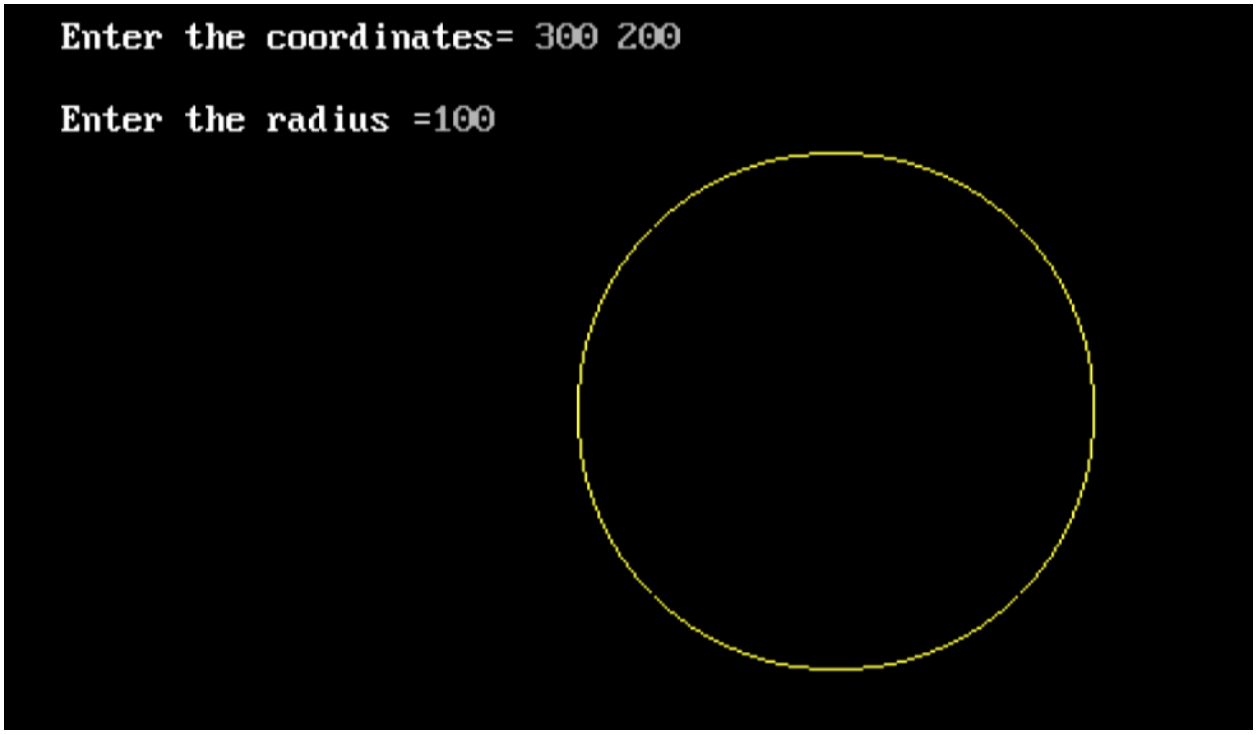
Source Code

```
#include <graphics.h>

#include <conio.h>

#include <stdio.h>


void main() {

   int x, y, x_mid, y_mid, radius, dp;


   int g_mode, g_driver = DETECT;

   initgraph(&g_driver, &g_mode, "C:\\Turboc3\\BGI");


   // Input center coordinates and radius from the user

   printf("Enter the coordinates: ");

   scanf("%d %d", &x_mid, &y_mid);


   printf("Enter the radius: ");

   scanf("%d", &radius);


   x = 0;

   y = radius;

   dp = 1 - radius;
```

```
do {
  // Plot points in all octants using symmetry

   putpixel(x_mid + x, y_mid + y, YELLOW);
   putpixel(x_mid + y, y_mid + x, YELLOW);
   putpixel(x_mid - y, y_mid + x, YELLOW);
   putpixel(x_mid - x, y_mid + y, YELLOW);
   putpixel(x_mid - x, y_mid - y, YELLOW);
   putpixel(x_mid - y, y_mid - x, YELLOW);
   putpixel(x_mid + y, y_mid - x, YELLOW);
   putpixel(x_mid + x, y_mid - y, YELLOW);

   if (dp < 0) {
      dp += (2 * x) + 1;
   } else {
      y = y - 1;
      dp += (2 * x) - (2 * y) + 1;
   }

   x = x + 1;
} while (y > x);

getch();
```

```
    closegraph();


    return 0;
}
```

Output



Practical-8

Aim: Write a program to draw a circle using Bresenham's Circle Drawing Algorithm.

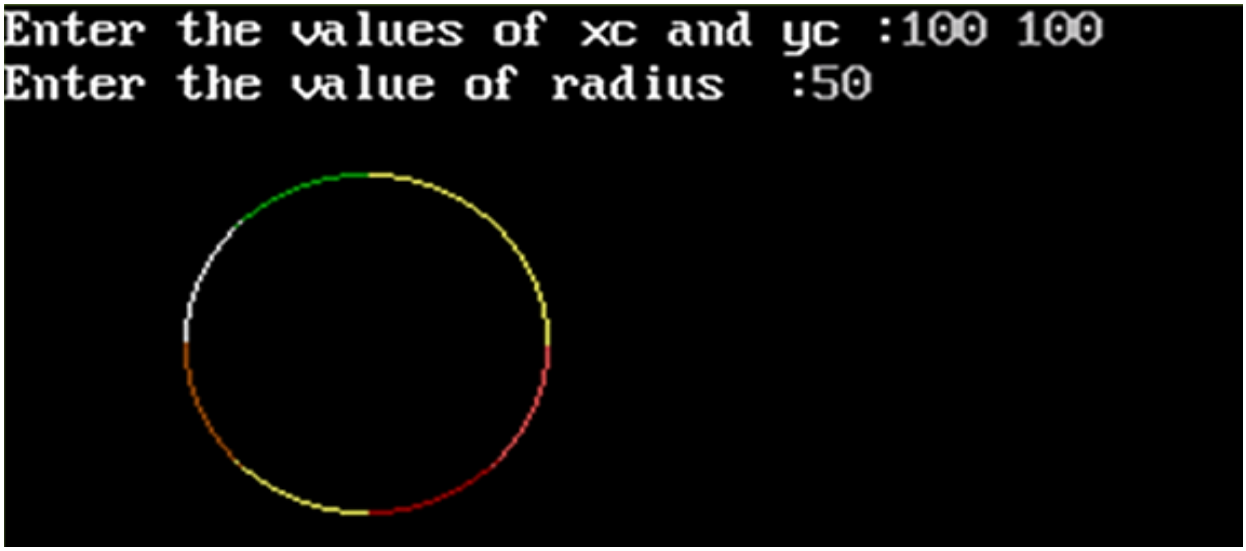
Source Code

```
#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <math.h>


// Used to plot pixels in all eight octants of the circle

void EightWaySymmetricPlot(int xc, int yc, int x, int y) {

   putpixel(x + xc, y + yc, RED);

   putpixel(x + xc, -y + yc, YELLOW);

   putpixel(-x + xc, -y + yc, GREEN);

   putpixel(-x + xc, y + yc, YELLOW);

   putpixel(y + xc, x + yc, 12);

   putpixel(y + xc, -x + yc, 14);

   putpixel(-y + xc, -x + yc, 15);

   putpixel(-y + xc, x + yc, 6);

}


void BresenhamCircle(int xc, int yc, int r) {
```

```
    int x = 0, y = r, d = 3 - (2 * r);
    EightWaySymmetricPlot(xc, yc, x, y);

    while (x <= y) {
        if (d <= 0) {
            d = d + (4 * x) + 6;
        } else {
            d = d + (4 * x) - (4 * y) + 10;
            y = y - 1;
        }
        x = x + 1;
        EightWaySymmetricPlot(xc, yc, x, y);
    }
}

int main(void) {
    int xc, yc, r, gdriver = DETECT, gmode;
    initgraph(&gdriver, &gmode, "C:\\TurboC3\\BGI");

    // Input the center coordinates and radius from the user
    printf("Enter the values of xc and yc: ");
    scanf("%d%d", &xc, &yc);
    printf("Enter the value of radius: ");
```

```
    scanf("%d", &r);
    BresenhamCircle(xc, yc, r);

    getch();
    closegraph();

    return 0;
}
```

Output



Practical-9

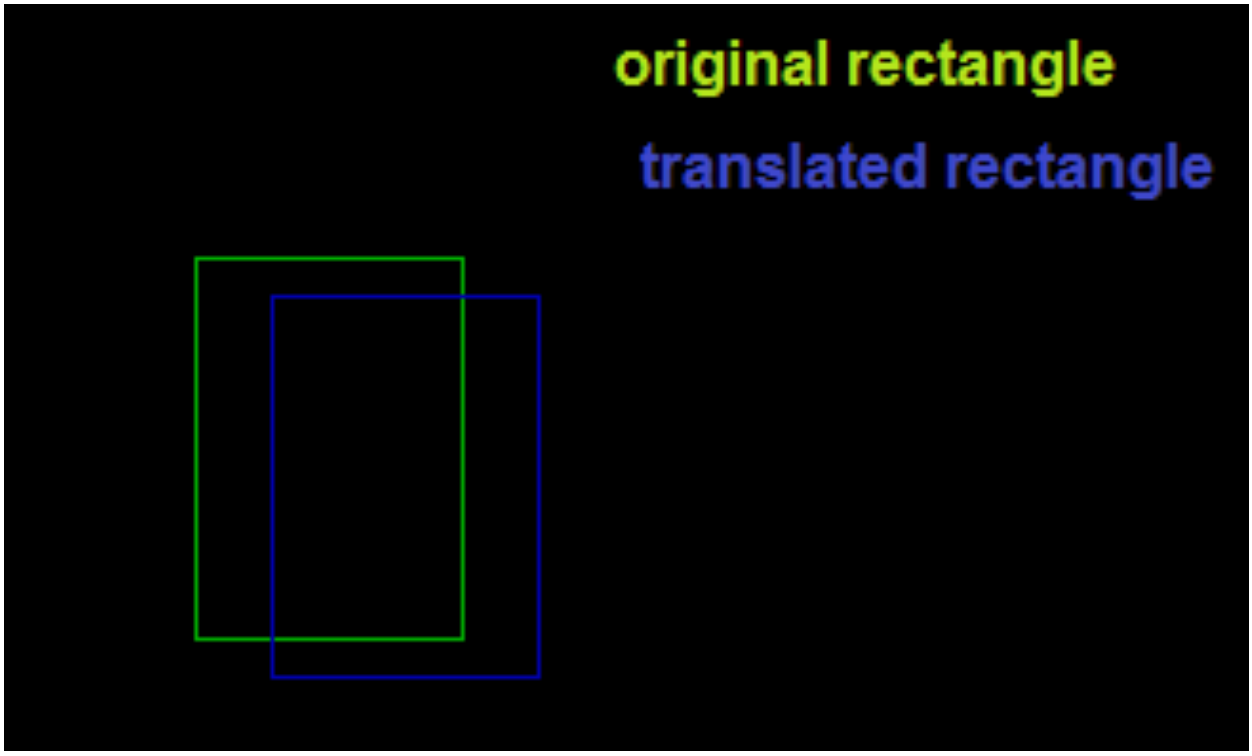
Aim: Write a program to translate an object and plot it at a new position.

Source Code

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

// Function to translate a rectangle by a given translation vector T
void translateRectangle ( int P[][2], int T[])
{
        int gd = DETECT, gm, errorcode;
        initgraph (&gd, &gm, "C:\\Turboc3\\BGI");
        setcolor (2);
        rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);   // Original rectangle

        // Translate the rectangle by adding the translation vector to its
        coordinates
        P[0][0] = P[0][0] + T[0];
        P[0][1] = P[0][1] + T[1];
        P[1][0] = P[1][0] + T[0];
        P[1][1] = P[1][1] + T[1];

        rectangle (P[0][0], P[0][1], P[1][0], P[1][1]);   // Translated rectangle
```

```
        closegraph();
}
int main()
{
        int P[2][2] = {5, 8, 12, 18};          // Initial coordinates of the rectangle
        int T[] = {2, 1};                       // Translation vector

        // Call the translation function
        translateRectangle (P, T);

        return 0;
}
```

Output



Practical-10

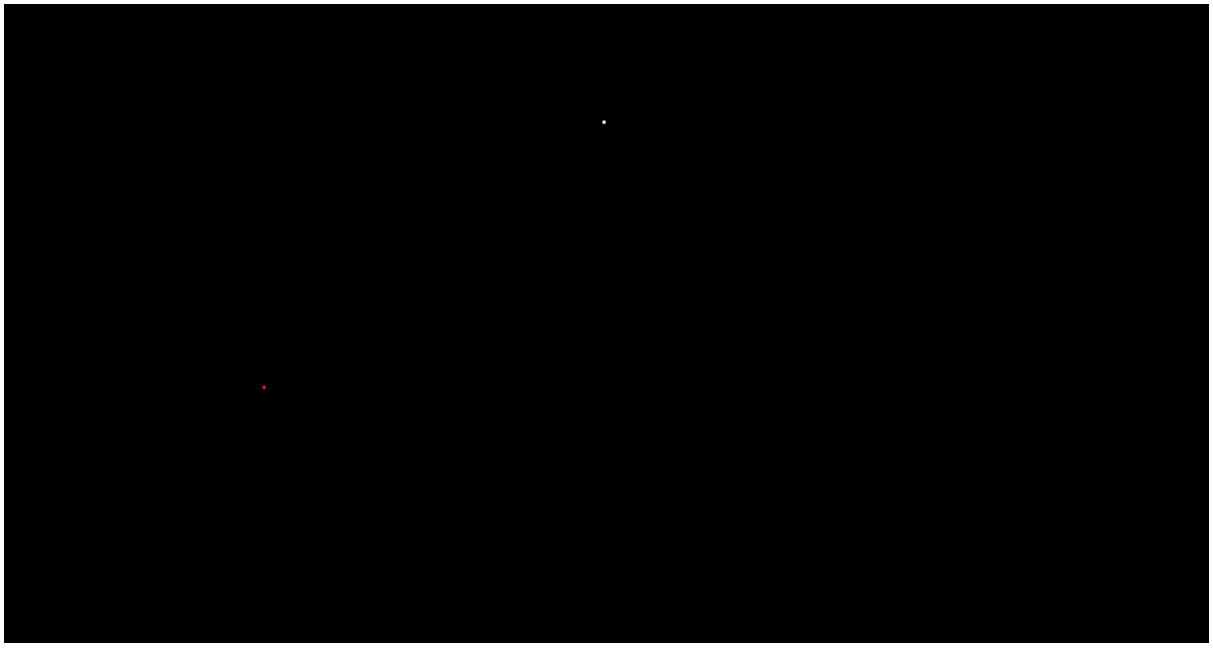
Aim: Write a program to rotate a point (100, 50) about origin in clockwise direction.

Source Code

```
#include <graphics.h>

#include <stdio.h>

#include <math.h>


// Function to rotate a point (x, y) about the origin in clockwise direction

void rotatePoint(int &x, int &y, float angle)

{

   float radianAngle = (angle * M_PI) / 180.0;

   int newX = (int)(x * cos(radianAngle) - y * sin(radianAngle));

   int newY = (int)(x * sin(radianAngle) + y * cos(radianAngle));

   x = newX;

   y = newY;

}


int main() {

   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");


   int x = 100, y = 50; // Coordinates of the original point
```

```
    putpixel(x, y, WHITE);
    float angle = 45.0; // Rotation angle in degrees (clockwise)

    // Rotate the point
    rotatePoint(x, y, angle);
    putpixel(x, y, RED);

    getch();
    closegraph();

    return 0;
}
```

Output



Practical-11

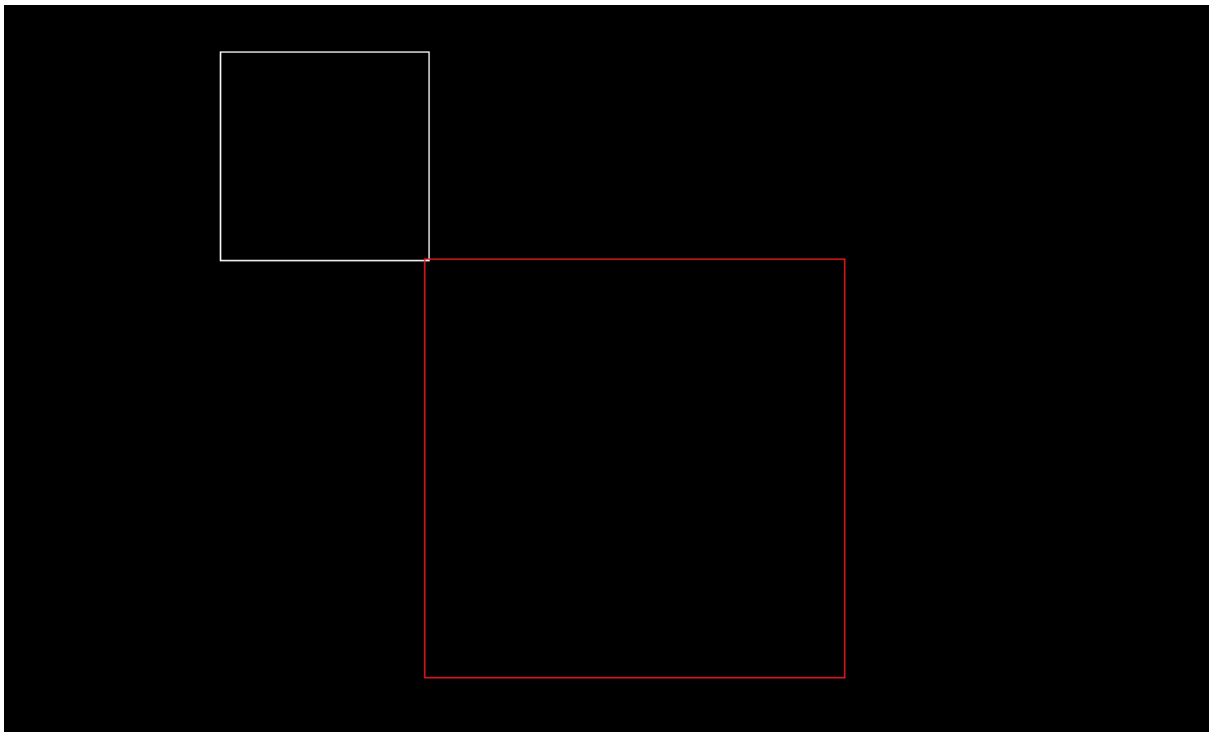
Aim: Write a program to scale an object to double of its size

Source Code

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

void drawRectangle(int x1, int y1, int x2, int y2) {
   rectangle(x1, y1, x2, y2);
}

// Function to scale an object to double its size
void scaleObject(int &x, int &y, float scaleFactor) {
   x = x * scaleFactor;
   y = y * scaleFactor;
}

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
   int x1 = 100, y1 = 100, x2 = 200, y2 = 200;
   drawRectangle(x1, y1, x2, y2);
   float scaleFactor = 2.0;                    // Scaling factor for doubling the size
```

```
    scaleObject(x1, y1, scaleFactor);
    scaleObject(x2, y2, scaleFactor);
    setcolor(RED);
    drawRectangle(x1, y1, x2, y2);

    getch();
    closegraph();

    return 0;
}
```

Output



Practical-12

Aim: Write a program to polygon using flood-fill method.

Source Code

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>


void flood(int x, int y, int new_col, int old_col) {
	// check current pixel is old_color or not
	if (getpixel(x, y) == old_col) {
		putpixel(x, y, new_col);
		flood(x + 1, y, new_col, old_col);
		flood(x - 1, y, new_col, old_col);
		flood(x, y + 1, new_col, old_col);
		flood(x, y - 1, new_col, old_col);
	}
}


int main() {
	int gd, gm = DETECT;
	initgraph(&gd, &gm, "");
	int top, left, bottom, right;
```

```
        top = left = 50;
        bottom = right = 300;
        rectangle(left, top, right, bottom);

        int x = 51;
        int y = 51;
        int newcolor = 12;
        int oldcolor = 0;

        flood(x, y, newcolor, oldcolor);
        getch();

        return 0;
}
```

Output

Practical-13

Aim: Write a program to fill a polygon using boundary-fill method.

Source Code

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

// Function for 8-connected Pixels
void boundaryFill8(int x, int y, int fill_color, int boundary_color) {
   if (getpixel(x, y) != boundary_color && getpixel(x, y) != fill_color) {
      putpixel(x, y, fill_color);
      boundaryFill8(x + 1, y, fill_color, boundary_color);
      boundaryFill8(x, y + 1, fill_color, boundary_color);
      boundaryFill8(x - 1, y, fill_color, boundary_color);
      boundaryFill8(x, y - 1, fill_color, boundary_color);
      boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
      boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
      boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
      boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
   }
}

int main()
```

```
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");

   rectangle(50, 50, 100, 100);
   boundaryFill8(55, 55, 4, 15);                    // Apply boundary fill algorithm

   getch();
   closegraph();

   return 0;
}
```

Output