INPUT

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>
using namespace std;
int a, b, Dx, Dy, temp, interchange, e; //Specifying parameters for algorithms
 int s1, s2;

void myInit() { //Initialization of viewing parameters
 glClearColor(1.0, 1.0, 1.0, 0.0);
 glMatrixMode(GL_PROJECTION);
 gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}

int sign(int a) { //Function to determine sign of a number
if (a > 0) {
 return 1;
}
 else {
  return -1;
 }
}

void plot(int a, int b) { //Function to plot points
 glBegin(GL_POINTS);
 glVertex2i(a, b);
 glEnd();
 glFlush();
}
#include<GL/glut.h>
```

```
void DrawPolygon(int x, int y, int m, int n) { //Function to draw polygon using DDA algorithm
 a = x; b = y;
 Dx = abs(m - x);
 Dy = abs(n - y);
 s1 = sign(m - x);
 s2 = sign(n - y);
 if (Dy > Dx) {
   temp = Dx;
   Dx = Dy;
   Dy = temp;
 interchange = 1;
 }
 else {
  interchange = 0;
 }

 e = 2 * Dy - Dx;
 plot(a, b);

 for (int i = 1; i <= Dx; i++) {
    plot(a, b);
    while (e >= 0) {
     if (interchange == 1)
       a = a + s1;
     else
       b = b + s2;
     e = e - 2 * Dx;
  }
  if (interchange == 1) {
    b = b + s2;
```

```
 }
 else {
  a = a + s1;
 }
 e = e + 2 * Dy;
}
}

 void putpixel(int c, int d, float* fillColor) { //Function to color polygon
 glColor3f(fillColor[0], fillColor[1], fillColor[2]); //Setting the color buffer
 glBegin(GL_POINTS);
 glVertex2i(c, d);
 glEnd();
 glFlush();
}

 void boundaryFill4(int x, int y, float* fillColor, float* boundarycolor) { //Boundary fill algorithm
 float color[3];
 glReadPixels(x, y, 1.0, 1.0, GL_RGB, GL_FLOAT, color);
  if ((color[0] != boundarycolor[0] || color[1] != boundarycolor[1] || color[2] !=
boundarycolor[2]) && (
color[0] != fillColor[0] || color[1] != fillColor[1] || color[2] != fillColor[2])) {
    putpixel(x, y, fillColor);
    boundaryFill4(x + 1, y, fillColor, boundarycolor); //Recurssive calls
          boundaryFill4(x - 2, y, fillColor, boundarycolor);
    boundaryFill4(x, y + 2, fillColor, boundarycolor);
          boundaryFill4(x, y - 2, fillColor, boundarycolor);
 }
}
```

```
void Floodfill4(int x, int y, float* fillColor, float* interiorColor) { //Floodfill algorithm
  float color[3];
  glReadPixels(x, y, 1.0, 1.0, GL_RGB, GL_FLOAT, color);
  if (color[0] == interiorColor[0] && color[1] == interiorColor[1] && color[2] == interiorColor[2]) {
    putpixel(x, y, fillColor);

          Floodfill4(x + 1, y, fillColor, interiorColor); //Recurssive calls
          Floodfill4(x - 1, y, fillColor, interiorColor);
       Floodfill4(x, y + 1, fillColor, interiorColor);
          Floodfill4(x, y - 1, fillColor, interiorColor);
  }
}

void myMouse(int button, int state, int x, int y) { //Function to select seed point
 y = 480 - y;
 if (button == GLUT_LEFT_BUTTON)
 {
  if (state == GLUT_DOWN)
  {
    float boundaryCol[] = { 0,1,0 };
    float fillcolor[] = { 1,0,1 };
    boundaryFill4(x, y, fillcolor, boundaryCol);
  }
 }
}

void myMouse1(int button, int state, int x, int y) {
 y = 480 - y;
 if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
   float interiorcolor[] = { 1,1,1 };
```

```
   float fillcolor[] = { 1,1,0 };
   Floodfill4(x, y, fillcolor, interiorcolor);
  }
 }

 void myDisplay() { //Draws polygon on the screen
  glLineWidth(3.0); glPointSize(2.0);
  glClear(GL_COLOR_BUFFER_BIT); glColor3f(0, 1, 0);
  DrawPolygon(100, 100, 200, 200);
  DrawPolygon(200, 200, 400, 200);
  DrawPolygon(400, 200, 400, 100);
  DrawPolygon(400, 100, 100, 100); glEnd();
  glFlush();
 }

 void myMenu(int item) { //Menu function
  if (item == 1) {
   glutMouseFunc(myMouse);
  }
  else if (item == 2) {
   glutMouseFunc(myMouse1);
  }
 }

 int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); //Initializing display mode for window
   glutInitWindowSize(640, 480);
   glutInitWindowPosition(200, 200);
   glutCreateWindow("Polygon Filling");
```

```
    glutDisplayFunc(myDisplay);
    glutCreateMenu(myMenu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutAddMenuEntry("Boundary fill",1);
    glutAddMenuEntry("Flood fill",2);
    myInit();
    glutMainLoop();
    return 0;
}
```

OUTPUT