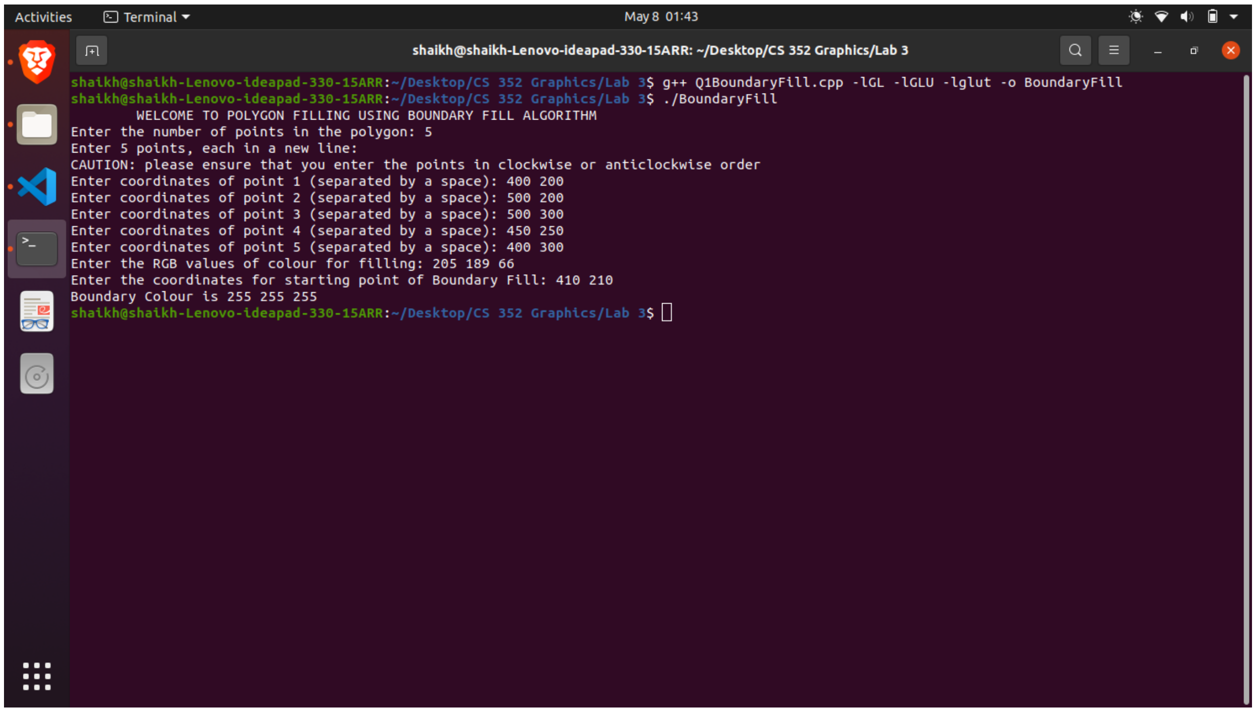
Name: Shaikh Ubaid
Roll no: 180001050
Date: 8 May, 2021
Assignment no: 3

Q1: Boundary Fill
Code:

```
#include <GL/glut.h> /* for using glut library */
#include <cstdio>    /* for using printf and scanf */
#include <vector>    /* for using vector */

#define WINDOW_WIDTH 1294
#define WINDOW_HEIGHT 704

// #define WINDOW_WIDTH 600
// #define WINDOW_HEIGHT 400

/* defining a structure for representing a point of the polygon */
typedef struct Point
{
   int x, y;
} Point;

/* defining a structure for representing a colour of a point */
typedef struct Colour
{
   unsigned char red, green, blue;
} Colour;

int no_of_polygon_points;
std::vector<Point> polygon;
Point starting_point;
Colour cur_pixel, fill_colour, boundary_colour;

/* declaring the functions */
void init(void);
void drawPolygon(std::vector<Point>);
int isColourEqual(Colour *c1, Colour *c2);
int isValid(int x, int y);
void putPixel(int x, int y);
void getPixel(int x, int y);
void boundaryFill(int x, int y);
void display(void);

int main(int argc, char **argv)
{
   printf("\tWELCOME TO POLYGON FILLING USING BOUNDARY FILL ALGORITHM\n"); /* showing welcome message */
   /* prompting the user for input */
   printf("Enter the number of points in the polygon: ");
   scanf("%d", &no_of_polygon_points);
   printf("Enter %d points, each in a new line:\n", no_of_polygon_points);
   printf("CAUTION: please ensure that you enter the points in clockwise or anticlockwise order\n");
   polygon.resize(no_of_polygon_points);
   for (int i = 0; i < no_of_polygon_points; ++i)
   {
       printf("Enter coordinates of point %d (separated by a space): ", i + 1);
       scanf("%d %d", &polygon[i].x, &polygon[i].y);
   }
   printf("Enter the RGB values of colour for filling: ");
   scanf("%d %d %d", (int *)&fill_colour.red, (int *)&fill_colour.green, (int *)&fill_colour.blue);
   printf("Enter the coordinates for starting point of Boundary Fill: ");
   scanf("%d %d", &starting_point.x, &starting_point.y);

   glutInit(&argc, argv);                                          /* initializing glut */
   glutInitDisplayMode(GLUT_SINGLE);                               /* use single color buffer and no depth
buffer */
   glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);                /* setting size of display area, in
pixels. */
   glutInitWindowPosition(0, 0);                                   /* setting location of window in screen
coordinates. */
   glutCreateWindow("POLYGON FILLING USING BOUNDARY FILL ALGORITHM"); /* giving name/title to window */
   init();                                                         /* setting our own OpenGL initialization
*/
   glutDisplayFunc(display);                                       /* register display() function as the
callback handler for window-paint event */
   glutMainLoop();                                                 /* run the event loop! This function
does not return */
   /* Program ends when user closes the window */
   return 0;
}

void init(void) /* initialize OpenGL Graphics */
{
   glClearColor(0.0, 0.0, 0.0, 1.0); /* set the "clearing" or background color as black and opaque*/
   glColor3ub(255, 255, 255);        /* set white colour as drawing colour */
   // glColor3i(255, 255, 255);
   glPointSize(1.0);                 /* each pixel is of size 1x1 */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT); /* setting window dimension in X and Y directions */
}

/* A function for drawing polygon. It expects given polygon points to be in clockwise or anticlockwise order
*/
void drawPolygon(std::vector<Point> polygon)
{
   glBegin(GL_LINE_LOOP); /* using line loop so that the polygon is not already filled */
   int N = polygon.size();
   for (int i = 0; i < N; ++i)
       glVertex2i(polygon[i].x, polygon[i].y);
   glEnd();
}

/* a function which returns true if given two colours are equal else it returns false */
int isColourEqual(Colour *c1, Colour *c2)
{
   return ((c1->red == c2->red) && (c1->green == c2->green) && (c1->blue == c2->blue));
}

/* a function which returns true if given point is inside the window */
int isValid(int x, int y)
{
   return !(x < 0 || y < 0 || x > WINDOW_WIDTH || y > WINDOW_HEIGHT);
}

/* a function to paint a pixel at given coordinates. It paints the pixel with colour value of fill_colour */
void putPixel(int x, int y)
{
   glRasterPos2i(x, y);
   glDrawPixels(1, 1, GL_RGB, GL_UNSIGNED_BYTE, &fill_colour);
   glFlush();
}
```

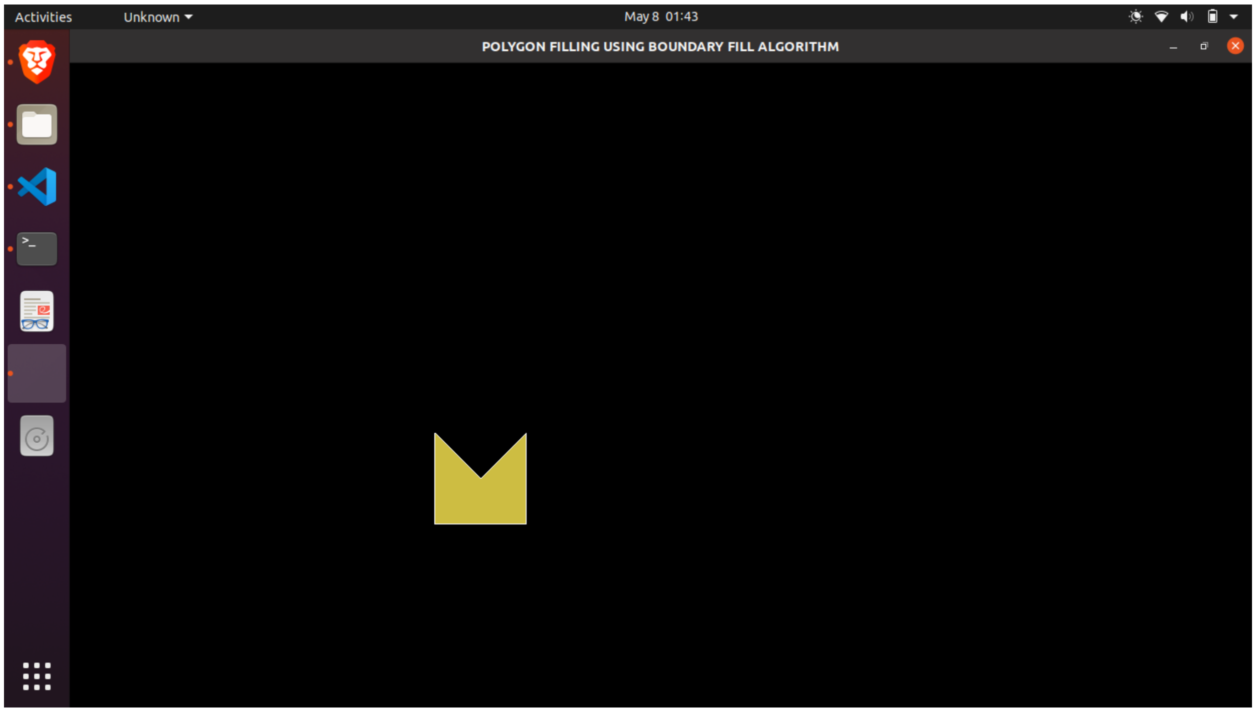
```c
/* a function to get the colour values at the given coordinates. It stores the colour values in cur_pixel */
void getPixel(int x, int y)
{
   glReadPixels(x, y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &cur_pixel);
}

/* my boundary fill function */
void boundaryFill(int x, int y)
{
   if(!isValid(x, y))
       return;

   getPixel(x, y);
   if (isColourEqual(&cur_pixel, &fill_colour) || isColourEqual(&cur_pixel, &boundary_colour))
       return;

   putPixel(x, y);
   boundaryFill(x - 1, y);
   boundaryFill(x + 1, y);
   boundaryFill(x, y - 1);
   boundaryFill(x, y + 1);
}

/* Our display function which runs when the window first appears and whenever there is a request to re-paint
the window. */
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT); /* clear the color buffer i.e. set background with the current "clearing"
color */

   drawPolygon(polygon); /* draw the polygon */

   glReadPixels(polygon[0].x, polygon[0].y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &boundary_colour); /* get the
colour of the boundary */

   /* printing boundary colour on the console for user verification */
   printf("Boundary Colour is %d %d %d\n", (int)boundary_colour.red, (int)boundary_colour.green,
(int)boundary_colour.blue);

   boundaryFill(starting_point.x, starting_point.y); /* fill the polygon */

   glFlush();
}
```

Input:


Activities
Terminal
May 8 01:43
shaikh@shaikh-Lenovo-ideapad-330-15ARR: ~/Desktop/CS 352 Graphics/Lab 3

```
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ g++ Q1BoundaryFill.cpp -lGL -lGLU -lglut -o BoundaryFill
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ ./BoundaryFill
        WELCOME TO POLYGON FILLING USING BOUNDARY FILL ALGORITHM
Enter the number of points in the polygon: 5
Enter 5 points, each in a new line:
CAUTION: please ensure that you enter the points in clockwise or anticlockwise order
Enter coordinates of point 1 (separated by a space): 400 200
Enter coordinates of point 2 (separated by a space): 500 200
Enter coordinates of point 3 (separated by a space): 500 300
Enter coordinates of point 4 (separated by a space): 450 250
Enter coordinates of point 5 (separated by a space): 400 300
Enter the RGB values of colour for filling: 205 189 66
Enter the coordinates for starting point of Boundary Fill: 410 210
Boundary Colour is 255 255 255
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ 
```

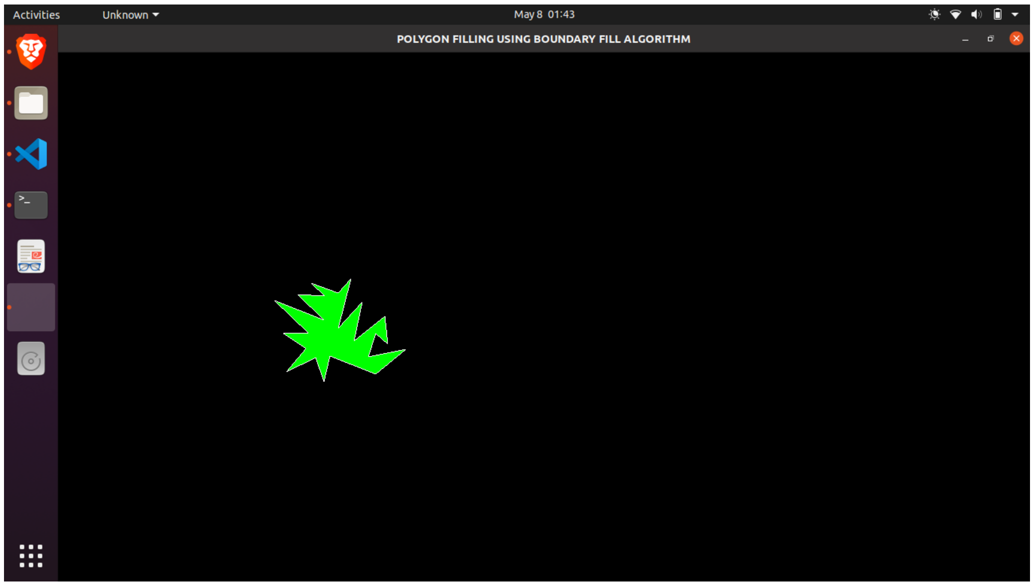
Output:



Test Case Input:

```
23
300 330
330 310
304 279
343 298
354 266
362 300
423 276
463 309
413 299
423 330
439 316
435 353
394 320
405 372
373 337
390 403
373 384
337 397
355 380
319 382
354 348
288 374
333 331
0 255 0
350 310
```

Test Case Output:



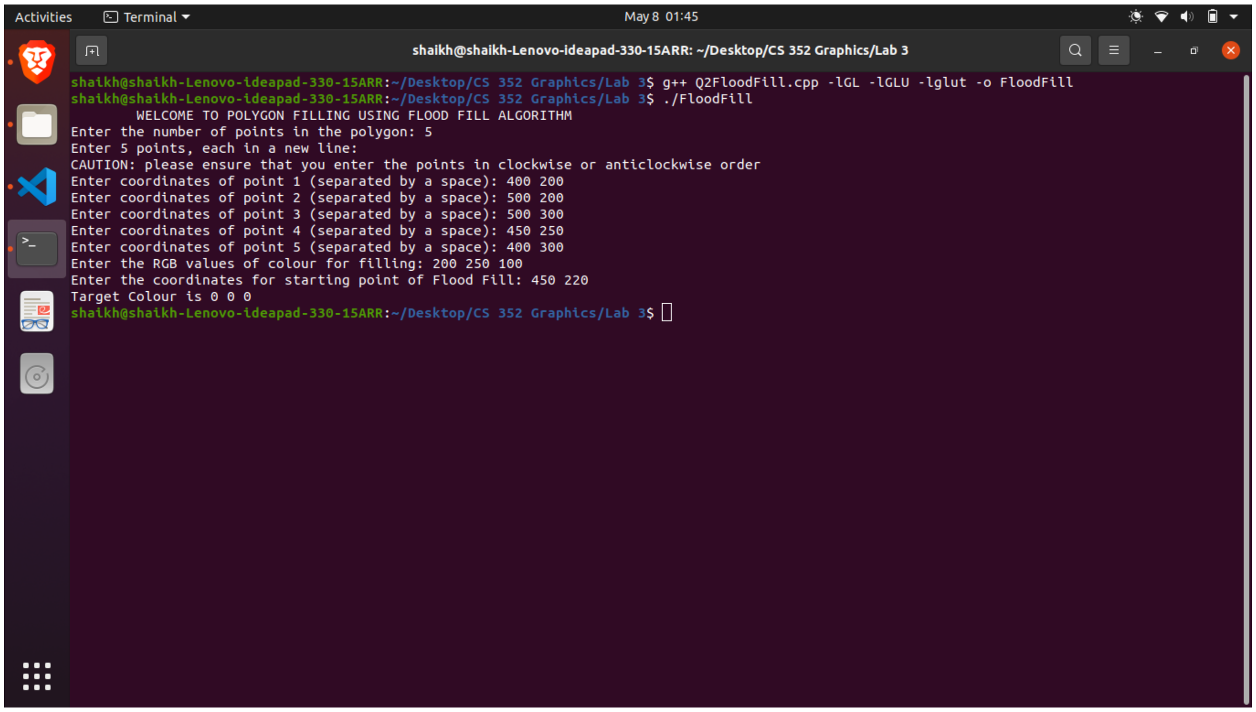
Q2: Flood Fill
Code:

```
#include <GL/glut.h> /* for using glut library */
#include <cstdio>    /* for using printf and scanf */
#include <vector>    /* for using vector */
#include <queue>     /* for using queue */

#define WINDOW_WIDTH 1294
#define WINDOW_HEIGHT 704

// #define WINDOW_WIDTH 600
// #define WINDOW_HEIGHT 400

/* defining a structure for representing a point of the polygon */
typedef struct Point
{
   int x, y;
} Point;

/* defining a structure for representing colour of a point */
typedef struct Colour
{
   unsigned char red, green, blue;
} Colour;

int no_of_polygon_points;
std::vector<Point> polygon;
Point starting_point;
Colour cur_pixel, replacement_colour, target_colour;

/* declaring the functions */
void init(void);
void drawPolygon(std::vector<Point>);
int isColourEqual(Colour *c1, Colour *c2);
int isValid(int x, int y);
void putPixel(int x, int y);
void getPixel(int x, int y);
void floodFill();
void display(void);

int main(int argc, char **argv)
{
   printf("\tWELCOME TO POLYGON FILLING USING FLOOD FILL ALGORITHM\n"); /* showing welcome message */
   /* prompting the user for input */
   printf("Enter the number of points in the polygon: ");
   scanf("%d", &no_of_polygon_points);
   printf("Enter %d points, each in a new line:\n", no_of_polygon_points);
   printf("CAUTION: please ensure that you enter the points in clockwise or anticlockwise order\n");
   polygon.resize(no_of_polygon_points);
   for (int i = 0; i < no_of_polygon_points; ++i)
   {
       printf("Enter coordinates of point %d (separated by a space): ", i + 1);
       scanf("%d %d", &polygon[i].x, &polygon[i].y);
   }
   printf("Enter the RGB values of colour for filling: ");
   scanf("%d %d %d", (int *)&replacement_colour.red, (int *)&replacement_colour.green, (int
*)&replacement_colour.blue);

   printf("Enter the coordinates for starting point of Flood Fill: ");
   scanf("%d %d", &starting_point.x, &starting_point.y);

   glutInit(&argc, argv);                                      /* initializing glut */
```

```
    glutInitDisplayMode(GLUT_SINGLE);                             /* use single color buffer and no depth
buffer */
    glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);              /* setting size of display area, in pixels.
*/
    glutInitWindowPosition(0, 0);                                 /* setting location of window in screen
coordinates. */
    glutCreateWindow("POLYGON FILLING USING FLOOD FILL ALGORITHM"); /* giving name/title to window */
    init();                                                       /* setting our own OpenGL initialization */
    glutDisplayFunc(display);                                     /* register display() function as the
callback handler for window-paint event */
    glutMainLoop();                                               /* run the event loop! This function does
not return */
    /* Program ends when user closes the window */
    return 0;
}

void init(void) /* initialize OpenGL Graphics */
{
    glClearColor(0.0, 0.0, 0.0, 1.0); /* set the "clearing" or background color as black and opaque*/
    glColor3ub(255, 255, 255);        /* set white colour as drawing colour */
    // glColor3i(255, 255, 255);
    glPointSize(1.0);                 /* each pixel is of size 1x1 */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT); /* setting window dimension in X and Y directions */
}

/* A function for drawing polygon. It expects given polygon points to be in clockwise or anticlockwise order
*/
void drawPolygon(std::vector<Point> polygon)
{
    glBegin(GL_LINE_LOOP); /* using line loop so that the polygon is not already filled */
    int N = polygon.size();
    for (int i = 0; i < N; ++i)
        glVertex2i(polygon[i].x, polygon[i].y);
    glEnd();
}

/* a function which returns true if given two colours are equal else it returns false */
int isColourEqual(Colour *c1, Colour *c2)
{
    return ((c1->red == c2->red) && (c1->green == c2->green) && (c1->blue == c2->blue));
}

/* a function which returns true if given point is inside the window */
int isValid(int x, int y)
{
    return !(x < 0 || y < 0 || x > WINDOW_WIDTH || y > WINDOW_HEIGHT);
}

/* a function to paint a pixel at given coordinates. It paints the pixel with colour value of fill_colour */
void putPixel(int x, int y)
{
    glRasterPos2i(x, y);
    glDrawPixels(1, 1, GL_RGB, GL_UNSIGNED_BYTE, &replacement_colour);
    glFlush();
}

/* a function to get the colour values at the given coordinates. It stores the colour values in cur_pixel */
void getPixel(int x, int y)
{
    glReadPixels(x, y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &cur_pixel);
}
```

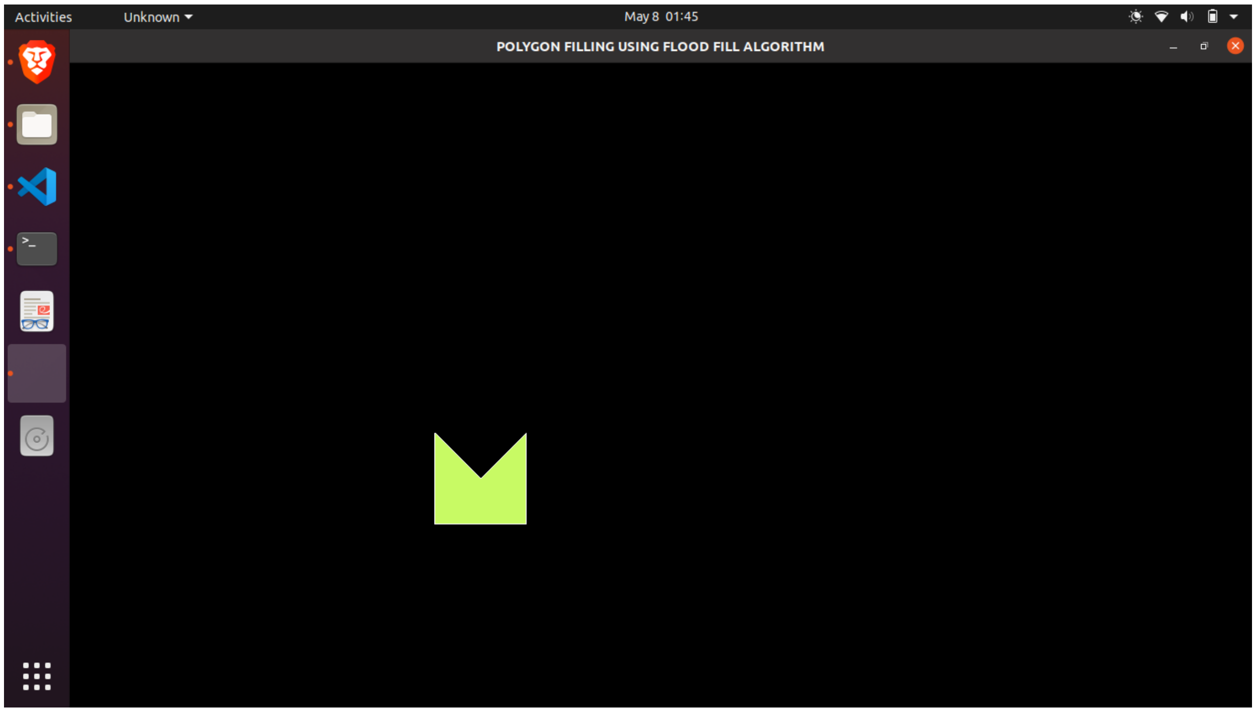
```
/* my floodfill function */
void floodFill()
{
   std::queue<Point> Q;
   Q.push(starting_point);
   while (!Q.empty()) /* while queue is not empty */
   {
       Point cur = Q.front();
       Q.pop();
       if (!isValid(cur.x, cur.y)) /* if the current point is invalid, go for next iteration */
           continue;

       getPixel(cur.x, cur.y);
       if (isColourEqual(&cur_pixel, &target_colour))
       {
           putPixel(cur.x, cur.y);
           Q.push({cur.x - 1, cur.y});
           Q.push({cur.x + 1, cur.y});
           Q.push({cur.x, cur.y - 1});
           Q.push({cur.x, cur.y + 1});
       }
   }
}

/* Our display function which runs when the window first appears and whenever there is a request to re-paint
the window. */
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT); /* clear the color buffer i.e. set background with the current "clearing"
color */

   drawPolygon(polygon); /* draw the polygon */

   glReadPixels(starting_point.x, starting_point.y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &target_colour); /* get
the colour of the points to be painted */

   /* printing target colour on the console for user verification */
   printf("Target Colour is %d %d %d\n", (int)target_colour.red, (int)target_colour.green,
(int)target_colour.blue);

   floodFill(); /* fill the polygon */

   glFlush();
}
```

Input:



```
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ g++ Q2FloodFill.cpp -lGL -lGLU -lglut -o FloodFill
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ ./FloodFill
        WELCOME TO POLYGON FILLING USING FLOOD FILL ALGORITHM
Enter the number of points in the polygon: 5
Enter 5 points, each in a new line:
CAUTION: please ensure that you enter the points in clockwise or anticlockwise order
Enter coordinates of point 1 (separated by a space): 400 200
Enter coordinates of point 2 (separated by a space): 500 200
Enter coordinates of point 3 (separated by a space): 500 300
Enter coordinates of point 4 (separated by a space): 450 250
Enter coordinates of point 5 (separated by a space): 400 300
Enter the RGB values of colour for filling: 200 250 100
Enter the coordinates for starting point of Flood Fill: 450 220
Target Colour is 0 0 0
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$
```

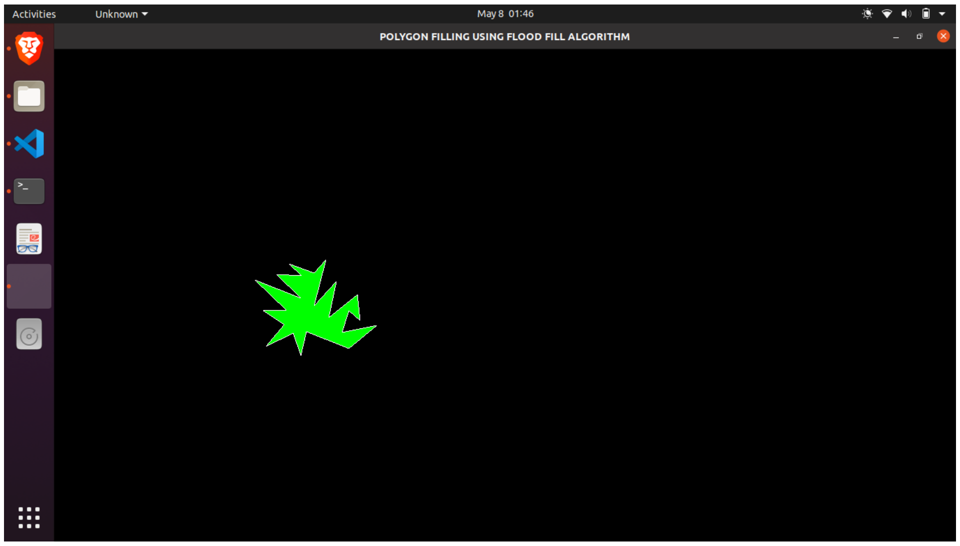
Output:



Test Case Input:

```
23
300 330
330 310
304 279
343 298
354 266
362 300
423 276
463 309
413 299
423 330
439 316
435 353
394 320
405 372
373 337
390 403
373 384
337 397
355 380
319 382
354 348
288 374
333 331
0 255 0
350 310
```

Test Case Output:



Q3: Scanline Seed Fill

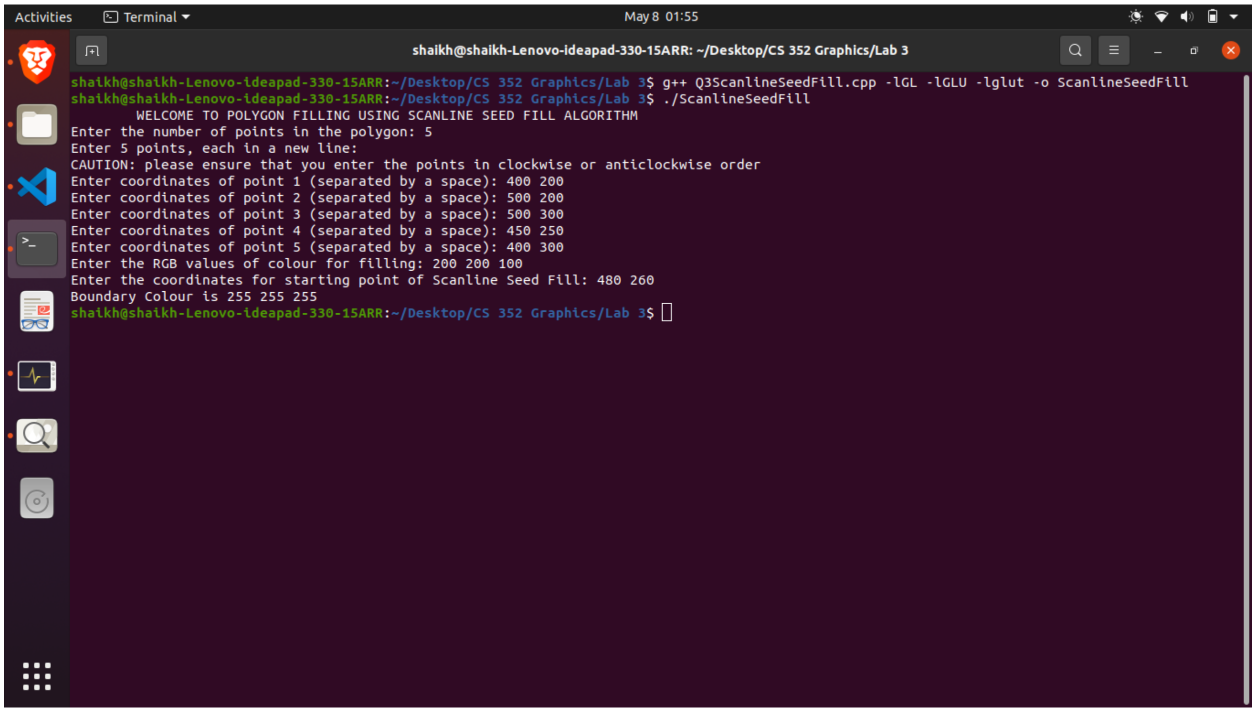
Code:

```
#include <GL/glut.h> /* for using glut library */
#include <cstdio>    /* for using printf and scanf */
#include <vector>    /* for using vector */
#include <stack>     /* for using stack */

#define WINDOW_WIDTH 1294
#define WINDOW_HEIGHT 704

// #define WINDOW_WIDTH 600
// #define WINDOW_HEIGHT 400

/* defining a structure for representing a point of the polygon */
typedef struct Point
{
   int x, y;
} Point;

/* defining a structure for representing colour of a point */
typedef struct Colour
{
   unsigned char red, green, blue;
} Colour;

int no_of_polygon_points;
std::vector<Point> polygon;
Point starting_point;
Colour cur_pixel, fill_colour, boundary_colour;

/* declaring the functions */
void init(void);
void drawPolygon(std::vector<Point>);
int isColourEqual(Colour *c1, Colour *c2);
int isValid(int x, int y);
void putPixel(int x, int y);
void getPixel(int x, int y);
int isBoundary(int x, int y);
int isFilled(int x, int y);
void scanlineSeedFill();
void display(void);

int main(int argc, char **argv)
{
   printf("\tWELCOME TO POLYGON FILLING USING SCANLINE SEED FILL ALGORITHM\n"); /* showing welcome message */
   /* prompting the user for input */
   printf("Enter the number of points in the polygon: ");
   scanf("%d", &no_of_polygon_points);
   printf("Enter %d points, each in a new line:\n", no_of_polygon_points);
   printf("CAUTION: please ensure that you enter the points in clockwise or anticlockwise order\n");
   polygon.resize(no_of_polygon_points);
   for (int i = 0; i < no_of_polygon_points; ++i)
   {
       printf("Enter coordinates of point %d (separated by a space): ", i + 1);
       scanf("%d %d", &polygon[i].x, &polygon[i].y);
   }
   printf("Enter the RGB values of colour for filling: ");
   scanf("%d %d %d", (int *)&fill_colour.red, (int *)&fill_colour.green, (int *)&fill_colour.blue);

   printf("Enter the coordinates for starting point of Scanline Seed Fill: ");
   scanf("%d %d", &starting_point.x, &starting_point.y);
   glutInit(&argc, argv);                                      /* initializing glut */
   glutInitDisplayMode(GLUT_SINGLE);                           /* use single color buffer and no depth
buffer */
   glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);            /* setting size of display area, in
pixels. */
   glutInitWindowPosition(0, 0);                               /* setting location of window in screen
coordinates. */
   glutCreateWindow("POLYGON FILLING USING SCANLINE SEED FILL ALGORITHM"); /* giving name/title to window */
   init();                                                     /* setting our own OpenGL initialization
*/
   glutDisplayFunc(display);                                   /* register display() function as the
callback handler for window-paint event */
   glutMainLoop();                                             /* run the event loop! This function
does not return */
   /* Program ends when user closes the window */
   return 0;
}


void init(void) /* initialize OpenGL Graphics */
{
   glClearColor(0.0, 0.0, 0.0, 1.0); /* set the "clearing" or background color as black and opaque*/
   glColor3ub(255, 255, 255);        /* set white colour as drawing colour */
   // glColor3i(255, 255, 255);
   glPointSize(1.0);                 /* each pixel is of size 1x1 */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT); /* setting window dimension in X and Y directions */
}


/* A function for drawing polygon. It expects given polygon points to be in clockwise or anticlockwise order
*/
void drawPolygon(std::vector<Point> polygon)
{
   glBegin(GL_LINE_LOOP);
   int N = polygon.size();
   for (int i = 0; i < N; ++i)
       glVertex2i(polygon[i].x, polygon[i].y);
   glEnd();
}

/* a function which returns true if given two colours are equal else it returns false */
int isColourEqual(Colour *c1, Colour *c2)
{
   return ((c1->red == c2->red) && (c1->green == c2->green) && (c1->blue == c2->blue));
}

/* a function which returns true if given point is inside the window */
int isValid(int x, int y)
{
   return !(x < 0 || y < 0 || x > WINDOW_WIDTH || y > WINDOW_HEIGHT);
}

/* a function to paint a pixel at given coordinates. It paints the pixel with colour value of fill_colour */
void putPixel(int x, int y)
{
   glRasterPos2i(x, y);
   glDrawPixels(1, 1, GL_RGB, GL_UNSIGNED_BYTE, &fill_colour);
   glFlush();
}

/* a function to get the colour values at the given coordinates. It stores the colour values in cur_pixel */
```

```
void getPixel(int x, int y)
{
   glReadPixels(x, y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &cur_pixel);
}

/* a function which returns true if the given coordinates are of a boundary point, else it returns false */
int isBoundary(int x, int y)
{
   getPixel(x, y);
   return isColourEqual(&cur_pixel, &boundary_colour);
}

/* a function which returns true if the given coordinates are filled with fill_colour, else it returns false
*/
int isFilled(int x, int y)
{
   getPixel(x, y);
   return isColourEqual(&cur_pixel, &fill_colour);
}

/* my scanline seed fill function */
void scanlineSeedFill()
{
   std::stack<Point> S;
   S.push(starting_point);
   while (!S.empty()) /* while stack is not empty */
   {
       Point cur = S.top();
       S.pop();

       putPixel(cur.x, cur.y);

       int x, y, xleft, xright;
       x = cur.x - 1;
       while (isValid(x, cur.y) && !isBoundary(x, cur.y))
       { /* fill as left as possible */
           putPixel(x, cur.y);
           x--;
       }
       xleft = x + 1; /* extreme left end */

       x = cur.x + 1;
       while (isValid(x, cur.y) && !isBoundary(x, cur.y))
       { /* fill as right as possible */
           putPixel(x, cur.y);
           x++;
       }
       xright = x - 1; /* extreme right end */


       /* find spans of unfilled points in the upper scanline */
       y = cur.y + 1;
       x = xleft;
       while (isValid(x, y) && x <= xright)
       {
           while (x <= xright && (isBoundary(x, y) || isFilled(x, y)))
           {
               x++;
           }
           int xprev = x;
           while (x <= xright && (!isBoundary(x, y) && !isFilled(x, y)))
           {
               xprev = x;
```

```
                x++;
            }
            if (xprev != x)
            {
                S.push({xprev, y}); /* push the extreme right end of the found span */
            }
        }


        /* find spans of unfilled points in the lower scanline */
        y = cur.y - 1;
        x = xleft;
        while (isValid(x, y) && x <= xright)
        {
            while (x <= xright && (isBoundary(x, y) || isFilled(x, y)))
            {
                x++;
            }
            int xprev = x;
            while (x <= xright && (!isBoundary(x, y) && !isFilled(x, y)))
            {
                xprev = x;
                x++;
            }
            if (xprev != x)
            {
                S.push({xprev, y}); /* push the extreme right end of the found span */
            }
        }
    }
}

/* Our display function which runs when the window first appears and whenever there is a request to re-paint
the window. */
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT); /* clear the color buffer i.e. set background with the current "clearing"
color */

   drawPolygon(polygon); /* draw the polygon */

   glReadPixels(polygon[0].x, polygon[0].y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, &boundary_colour); /* get the
colour of the boundary */

   /* printing boundary colour on the console for user verification */
   printf("Boundary Colour is %d %d %d\n", (int)boundary_colour.red, (int)boundary_colour.green,
(int)boundary_colour.blue);

   scanlineSeedFill(); /* fill the polygon */

   glFlush();
}
```

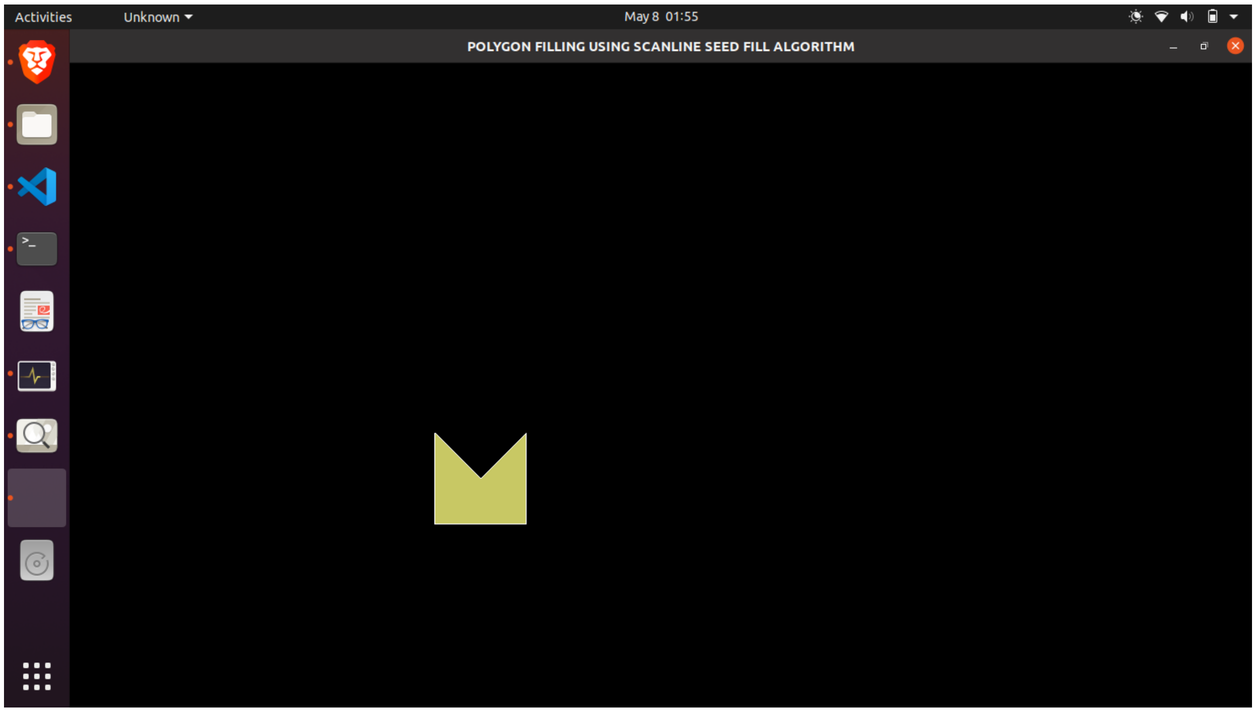
Input:



Activities
Terminal
May 8 01:55
shaikh@shaikh-Lenovo-ideapad-330-15ARR: ~/Desktop/CS 352 Graphics/Lab 3

```
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ g++ Q3ScanlineSeedFill.cpp -lGL -lGLU -lglut -o ScanlineSeedFill
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$ ./ScanlineSeedFill
        WELCOME TO POLYGON FILLING USING SCANLINE SEED FILL ALGORITHM
Enter the number of points in the polygon: 5
Enter 5 points, each in a new line:
CAUTION: please ensure that you enter the points in clockwise or anticlockwise order
Enter coordinates of point 1 (separated by a space): 400 200
Enter coordinates of point 2 (separated by a space): 500 200
Enter coordinates of point 3 (separated by a space): 500 300
Enter coordinates of point 4 (separated by a space): 450 250
Enter coordinates of point 5 (separated by a space): 400 300
Enter the RGB values of colour for filling: 200 200 100
Enter the coordinates for starting point of Scanline Seed Fill: 480 260
Boundary Colour is 255 255 255
shaikh@shaikh-Lenovo-ideapad-330-15ARR:~/Desktop/CS 352 Graphics/Lab 3$
```

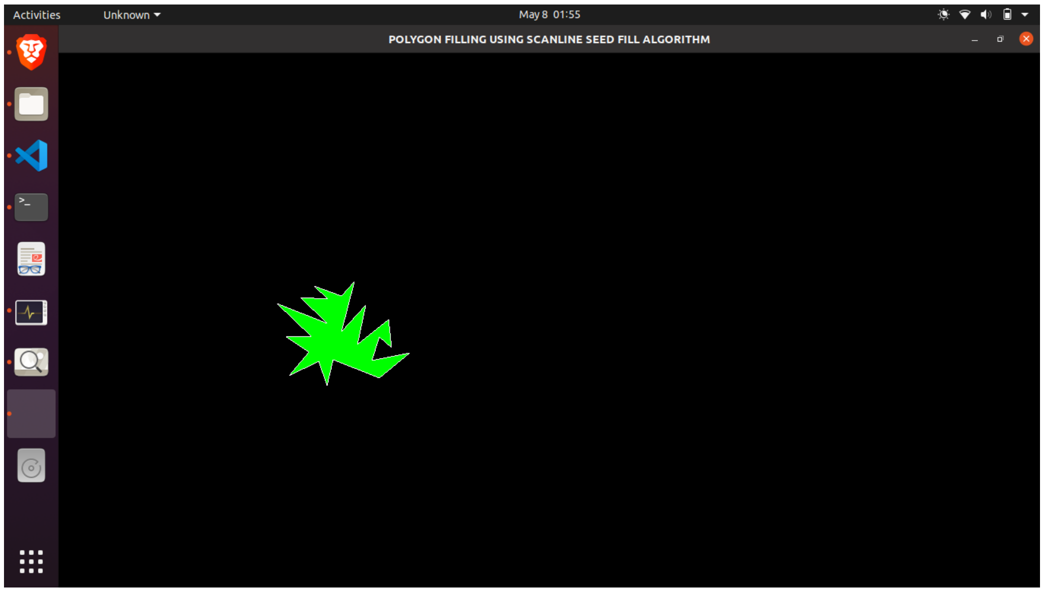
Output:



Test Case Input:

```
23
300 330
330 310
304 279
343 298
354 266
362 300
423 276
463 309
413 299
423 330
439 316
435 353
394 320
405 372
373 337
390 403
373 384
337 397
355 380
319 382
354 348
288 374
333 331
0 255 0
350 310
```

Test Case Output:


Activities
Unknown
May 8 01:55
POLYGON FILLING USING SCANLINE SEED FILL ALGORITHM