Logic

Algorithm

1. Input the two endpoints of line and store the left endpoint in (x_0, y_0)
2. Load (x_0, y_0) into framebuffer, that is, plot the first point
3. Calculate constants $\Delta x, \Delta y, 2\Delta y, 2\Delta y - 2\Delta x$ and obtain the starting value for decision parameter as p_0, where $p_0 = 2\Delta y - \Delta x$
4. At each x_k along the line, starting at $k=0$, perform:

   If $p_k < 0$, the next point to plot is (x_{k+1}, y_k) and $p_{k+1} = p_k + 2\Delta y$.

   else, the next point to plot is (x_{k+1}, y_{k+1}) and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Repeat step (4) Δx times

Date 30/10/20

Expt. No. 1 Page No.

Q. Write a program to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and slopes less than one. User must be able to draw as many lines and specify inputs through keyboard/mouse.

```
#include <GL/glut.h>
#include <stdio.h>
#include "header.h"
void display_p1(int x, int y)
{  glBegin(GL_POINTS);
   glVertex2i(x, y);
   glEnd();
}
int x1, y1, x2, y2;
void draw_line()
{ glClear(GL_COLOR_BUFFER_BIT);
  int dx, dy, i, e, incx, incy, inc1, inc2, x, y;
  dx = x2 - x1;  dy = y2 - y1;
  if (dx < 0) dx = -dx;
  if (dy < 0) dy = -dy;
  incx = 1;
  if (x2 < x1) incx = -1;
  incy = 1;
  if (y2 < y1) incy = -1;
  x = x1;  y = y1;
  if (dx > dy)
  {  display_p1(x, y);
     e = 2 * dx - dy;
     inc1 = 2 * (dy - dx);
```

Teacher's Signature

```
#include <iostream>
#include <GL/glut.h>
#include <time.h>
using namespace std;
int x1, x2, yc1, y2;
int flag1 = 0;
void draw_pixel(int x, int y)
{
        glColor3f(1, 0, 0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
        glFlush();

}
void draw_line()
{
        int dx, dy, i, e;
        int incx, incy, inc1, inc2;
        int x, y;
        dx = x2 - x1;
        dy = y2 - yc1;
        if (dx < 0)dx = -dx;
        if (dy < 0)dy = -dy;
        incx = 1;
        if (x2 < x1)
                incx = -1;
        incy = 1;
        if (y2 < yc1)
                incy = -1;
        x = x1;
        y = yc1;
        if (dx > dy)
        {
                draw_pixel(x, y);
                e = 2 * dy - dx;
                inc1 = 2 * (dy - dx);
                inc2 = 2 * dy;
                for (i = 0; i < dx; i++)
                {
                        if (e > 0)
                        {
                                y += incy;
                                e += inc1;
                        }
                        else
                                e += inc2;
                        x += incx;
                        draw_pixel(x, y);
                }
        }
        else
        {
                draw_pixel(x, y);
```

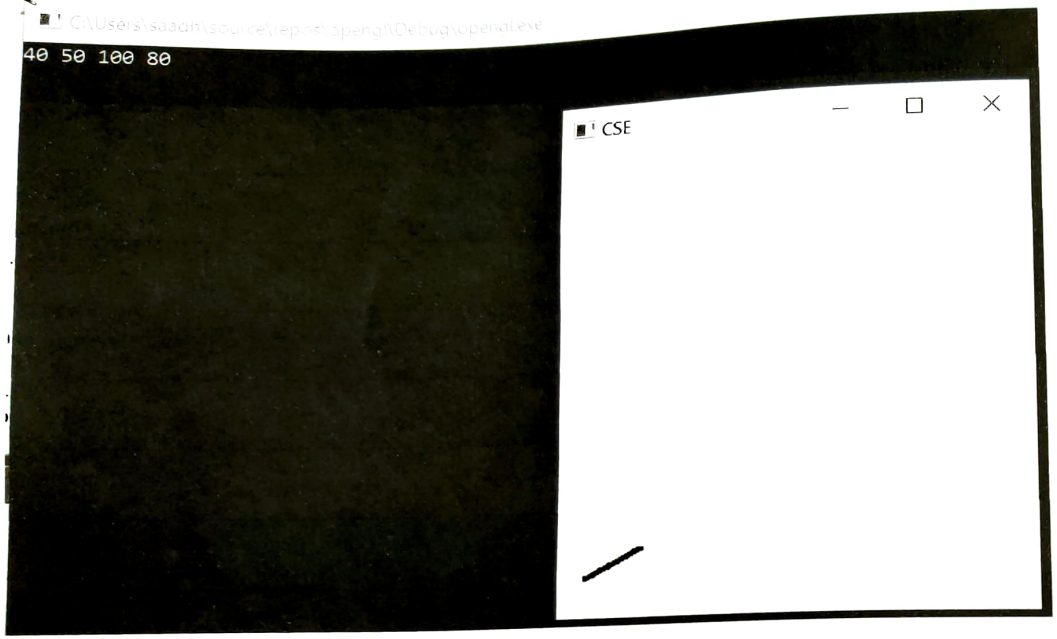
Date

Expt. No.

Page No.

```
inc2 = 2 * dy;
for ( i=0; i< dx ;i++)
{  if (e>=0)
    {   y+= incy;
        e += inc1;
    }
    else  e += inc2;
    x += incx;
    display-p1 (x, y);
  }
}
else
{ display-p1 (x, y);
   e = 2 * dy - dx;
   inc1 = 2 * (dy - dx);
   inc2 = 2 * dx;
   for ( i = 0 ; i < dy ; i++)
   {  if (e >= 0)
      {  x += incx;
         e += inc1;
      }
      else  e += inc2;
      y += incy;
      display-p1 (x, y);
   }
}
glFush();
}
```

Teacher's Signature



Date

Expt. No.

Page No.

```
void minit()
{ glClearColor(1,1,1,1);
  glColor3f(1.0, 0.0, 0.0);
  glPointSize(2.0);
  gluOrtho2D(10,500,10,500);
}
void PI_main(int argc, char*argv[])
{ scanf_s("%d %d %d %d", &x1, &y1, &x2, &y2);
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(300,300);
  glutInitWindowPosition(5,5);
  glutCreateWindow("CSE");
  glutDisplayFunc(draw_line);
  minit();
  glutMainLoop();
}
```

Teacher's Signature

Logic

A. Algorithm for circle drawing in Bresenham's algorithm.

1. Initialize x to 0, y to r and let decision parameter P_0 be $3-2r$.
2. While $x_i \geq y_i$, find parameter value such that

   If $P_i < 0$,

   y remains the same, x is incremented by 1 and next parameter is $P_i + 4x_i + 6 = P_{i+1}$

   else,

   y is decremented by 1, x is incremented by 1 and next parameter $P_{i+1} = P_i + 4(x_i - y_i) + 10$.
3. By symmetry calculate for the entire circle.
4. The points plotted for a circle with center (x_c, y_c) are $(x_c + x_k, y_c + y_k)$ in the octant calculated in the algorithm.

Date 30/10/20

Expt. No. 2

Page No.

Q. Write a program to generate a circle and ellipse using Bresenham's circle drawing and ellipse drawing techniques. Use two windows to draw circle in one window and ellipse in the other window. User can specify inputs through keyboard/mouse.

```
#include <GL/glut.h>
#include <stdio.h>
#include <math.h>
int yc, xc, r, rx, ry, xce, yce;
void draw_circle(int xc, int yc, int x, int y)
{   glBegin(GL_POINTS);
    glVertex2i(xc+x, yc+y);
    glVertex2i(xc-x, yc+y);
    glVertex2i(xc+x, yc-y);
    glVertex2i(xc-x, yc-y);
    glVertex2i(xc+y, yc+x);
    glVertex2i(xc-y, yc+x);
    glVertex2i(xc+y, yc-x);
    glVertex2i(xc-y, yc-x);
    glEnd();
}
void circle_bresc()
{ glClear(GL_COLOR_BUFFER_BIT);
  int x=0, y=r;
  int d = 3-2*r;
  while (x<=y)
  { draw_circle(xc, yc, x, y);
    if(d<0) d = d + 4*x + 6;
```

Teacher's Signature

8. Algorithm for ellipse drawing with midpoint ellipse algorithm

1. The first point on ellipse centred on origin is $(x_0, y_0) = (0, r_y)$
2. Calculate initial parameter ~~d~~ in region 1 as
$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$
3. At each x_i, starting from $i=0$, if $P1_i < 0$, next point is (x_{i+1}, y_i) and $P1_{i+1} = P1_i + 2r_y^2 x_{i+1} + r_y^2$
else, next point is $(x_{i+1}, y_i - 1)$ and $P1_{i+1} = P1_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_y^2$
4. Repeat step 3 till $2r_y^2 x \geq 2r_x^2 y$
5. Let last point calculated above be (x_0, y_0) Then initial parameter in region 2, $P2_0 = r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$.
6. At each y_i, starting from $i=0$, if $P2_i > 0$, next point is $(x_i, y_i - 1)$ and $P2_{i+1} = P2_i - 2r_x^2 r_{y_{i+1}} + r_x^2$
else, next point is $(x_i + 1, y_i - 1)$ and $P2_{i+1} = P2_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_x^2$
7. Repeat step 6 till $y=0$
8. Calculate points in the other three quandrants by symmetry.
7. If center is (x_c, y_c) instead of origin, plot $(x + x_c, y + y_c)$ for each calculated pixel position.

Date _____

Expt. No. _____ Page No. _____

```
        else
        { d = d + 4 * (x - y) + 10;
          y--;
        }
        x++;
    }
    glFush();
}
void draw_ellipse(int xce, int yce, int x, int y)
{   glBegin(GL_POINTS);
    glVertex2i(x + xce, y + yce);
    glVertex2i(-x + xce, y + yce);
    glVertex2i(x + xce, -y + yce);
    glVertex2i(-x + xce, -y + yce);
    glEnd();
}
void mid_point_ellipse()
{   glClear(GL_COLOR_BUFFER_BIT);
    float dx, dy, d1, d2, x = 0, y = ry;
    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
    while (dx < dy)
    {   draw_ellipse(xce, yce, x, y);
        if (d1 < 0)
        {   x++;
            dx = dx + (2 * ry * ry);
            d1 = d1 + dx + (ry * ry);
        }
```

Teacher's Signature _____

p2 - 1

```
#include<GL/glut.h>
#include<stdio.h>
#include<math.h>

int xc, yc, r;
int rx, ry, xce, yce;

// plots the points of a circle according to 8-fold symmetry
void draw_circle(int xc, int yc, int x, int y) {
        glBegin(GL_POINTS);
        glVertex2i(xc + x, yc + y);
        glVertex2i(xc - x, yc + y);
        glVertex2i(xc + x, yc - y);
        glVertex2i(xc - x, yc - y);
        glVertex2i(xc + y, yc + x);
        glVertex2i(xc - y, yc + x);
        glVertex2i(xc + y, yc - x);
        glVertex2i(xc - y, yc - x);
        glEnd();
}

// implements Bresenham's circle drawing algorithm
void circle_bres() {
        glClear(GL_COLOR_BUFFER_BIT);
        int x = 0, y = r;

        //initial decision parameter
        int d = 3 - 2 * r;
        while (x <= y) {
                draw_circle(xc, yc, x, y);
                if (d < 0)
                        d = d + 4 * x + 6;
                else {
                        d = d + 4 * (x - y) + 10;
                        y--;
                }
                x++;
        }
        glFlush();
}

/*
int p1_x, p2_x, p1_y, p2_y;
int point1_done = 0;
void myMouseFunccircle(int button, int state, int x, int y)
{
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN &&
point1_done == 0)
        {
                p1_x = x - 250;
                p1_y = 250 - y;
                point1_done = 1;
        }
        else if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
```

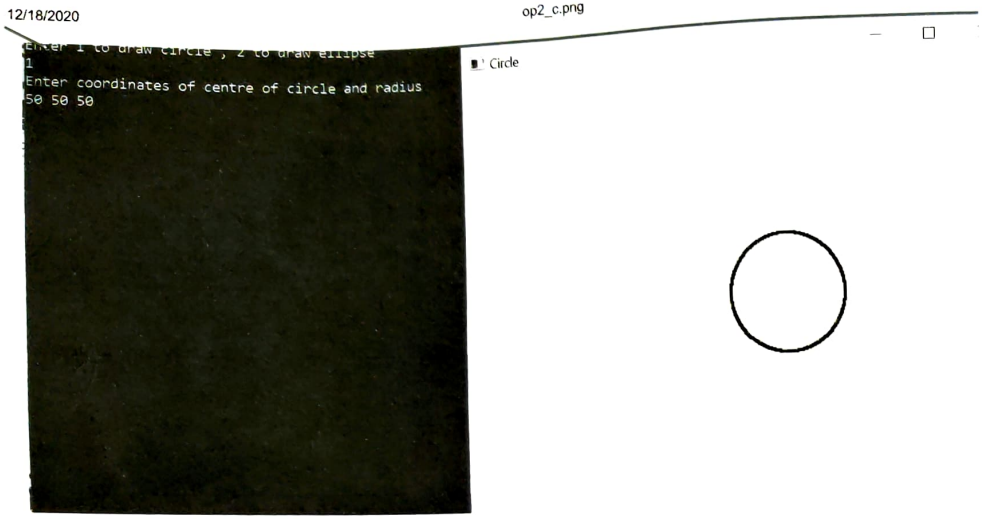
Date

Expt. No. Page No.

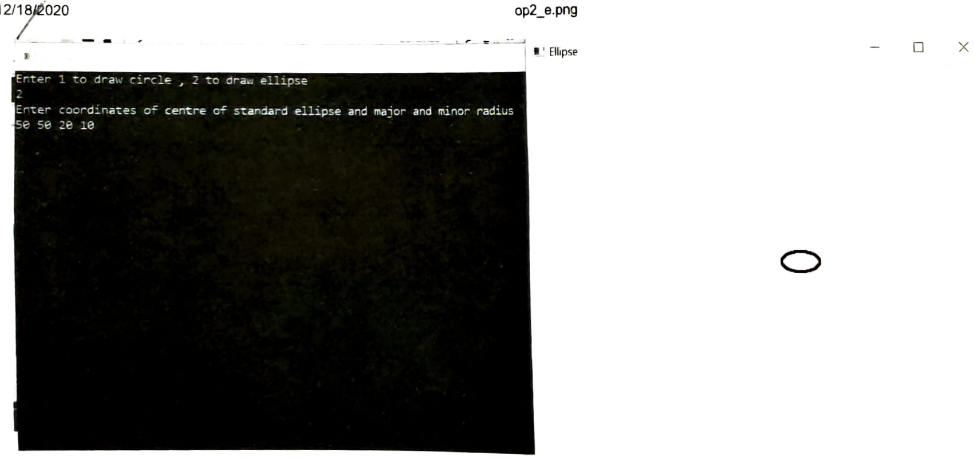
```
else
{ x++; y--;
  dx = dx + (2*ry*ry);
  dy = dy - (2*rx*rx);
  d1 = d1 + dx - dy + (ry*ry);
}
}
d2 = ((ry*ry)*((x+0.5)*(x+0.5))) + ((rx*rx)*
     ((y-1)*(y-1))) - (rx*rx*ry*ry);
while (y>=0)
{ draw_ellipse(xce,yce, x, y);
  if (d2>0)
  { y--;
    dy = dy - (2*rx*rx);
    d2 = d2 + (rx*rx) - dy;
  }
  else
  { y--; x++;
    dx = dx + (2*ry*ry);
    dy = dy - (2*rx*rx);
    d2 = d2 + dx - dy + (rx*rx);
  }
}
glFlush();
}
void minit2c()
{ glClearColor(1,1,1,1);
  glColor3f(1.0, 0.0, 0.0);
  glPointSize(3.0);
```

Teacher's Signature

12/18/2020 op2_c.png



12/18/2020 op2_e.png




```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-250,250,-250,250);
}
void pz_main(int argc, char** argv)
{ glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(500,500);
  glutInitWindowPosition(0,0);
  printf("Enter 1 to draw circle, 2 to draw ellipse\n");
  int ch;
  scanf-s("%d", &ch);
  switch(ch)
  { case 1: printf("Enter coordinates of center of circle and radius\n");
            scanf-s("%d %d %d", &xc, &yc, &r);
            glutCreateWindow("circle");
            glutDisplayFunc(circle_bres);
            break;
    case 2: printf("Enter coordinates of center of standard
            ellipse and major and minor radius\n");
            scanf-s("%d %d %d %d", &xce, &yce, &rx, &ry");
            glutCreateWindow("Ellipse");
            glutDisplayFunc(mid_point_ellipse);
            break;
  }
  minit2();
  glutMainLoop();
}
```

Date

Expt. No.

Page No.

Teacher's Signature

```
#include<iostream>
using namespace std;
#include<GL/glut.h>
#include "Header.h"

int m;
float tetra[4][3] = { { 0, 200, 400 }, { 0, 0, -350 }, { 200, 350, 300
}, { -300, 300, 200 } };

void draw_triangle(float p1[], float p2[], float p3[])
{
        glBegin(GL_TRIANGLES);
        glVertex3f(p1[0], p1[1], p1[2]);
        glVertex3f(p2[0], p2[1], p2[2]);
        glVertex3f(p3[0], p3[1], p3[2]);
        glEnd();
}

void divide_triangle(float a[], float b[], float c[], int m)
{

        float v1[3], v2[3], v3[3];
        int j;

        if (m > 0)
        {
                for (j = 0; j < 3; j++)
                        v1[j] = (a[j] + b[j]) / 2;
                for (j = 0; j < 3; j++)
                        v2[j] = (a[j] + c[j]) / 2;
                for (j = 0; j < 3; j++)
                        v3[j] = (c[j] + b[j]) / 2;
                divide_triangle(a, v1, v2, m - 1);
                divide_triangle(c, v2, v3, m - 1);
                divide_triangle(b, v3, v1, m - 1);
        }
        else
                draw_triangle(a, b, c);
}
void tetrahedron()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glColor3f(1.0, 0, 0);
        divide_triangle(tetra[0], tetra[1], tetra[2], m);

        glColor3f(0, 1.0, 0);
        divide_triangle(tetra[3], tetra[2], tetra[1], m);

        glColor3f(0, 0, 1.0);
        divide_triangle(tetra[0], tetra[3], tetra[1], m);

        glColor3f(0, 0, 0);
        divide_triangle(tetra[0], tetra[2], tetra[3], m);

        glFlush();
```

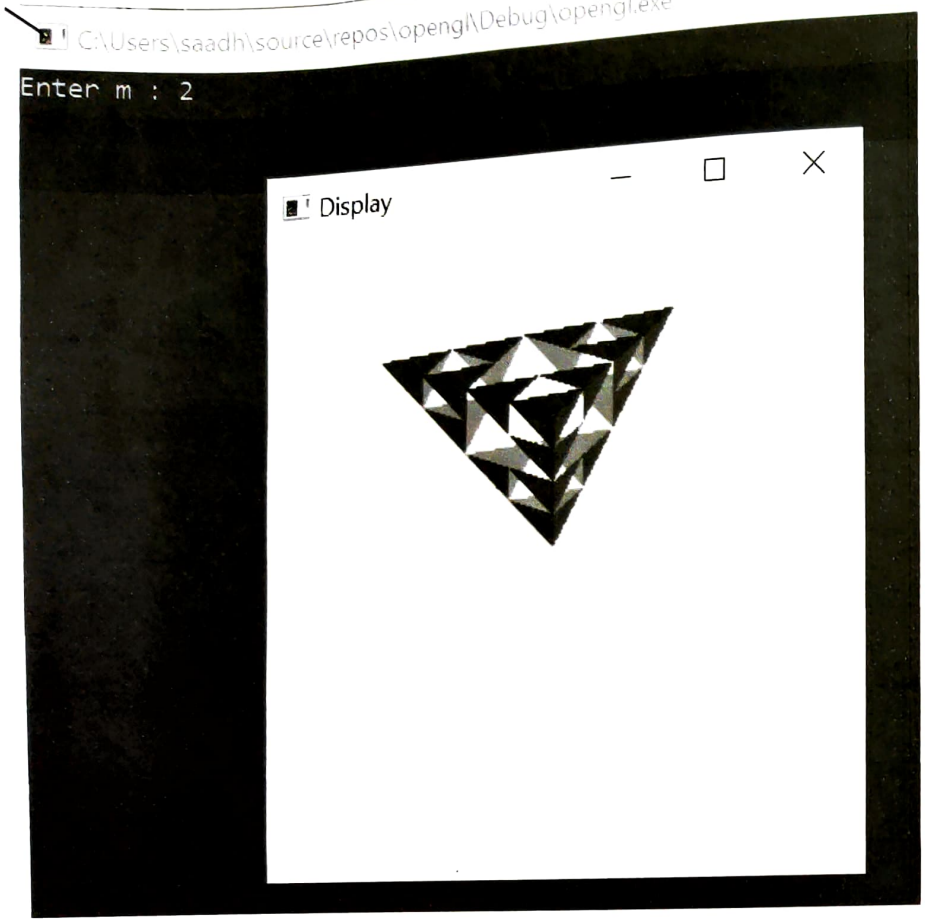
Date 30/10/20

Expt. No. 3 Page No.

8. Write a program to recursively subdivide a tetrahedron to form 3D Sierpinski gasket. The number of recursive steps is to be specified at execution time.

```
#include<iostream>
using namespace std;
#include<GL/glut.h>
#include "Header.h"
int m;
float tetra[4][3] = {{0,200,400}, {0, 0, -350}, {200, 350, 300},
    {-300, 300, 200}};
void draw_triangle(float p1[], float p2[], float p3[])
{ glBegin(GL_TRIANGLES);
  glVertex3f(p1[0], p1[1], p1[2]);
  glVertex3f(p2[0], p2[1], p2[2]);
  glVertex3f(p3[0], p3[1], p3[2]);
  glEnd();
}
void divide_triangle(float a[], float b[], float c[], int m)
{ float v1[3], v2[3], v3[3];
  int j;
  if(m>0)
  { for(j=0; j<3; j++) v1[j] = (a[j] + b[j])/2;
    for(j=0; j<3; j++) v2[j] = (a[j] + c[j])/2;
    for(j=0; j<3; j++) v3[j] = (b[j] + c[j])/2;
    divide_triangle(a, v1, v2, m-1);
    divide_triangle(c, v2, v3, m-1);
    divide_triangle(b, v3, v1, m-1);
  }
```

Teacher's Signature



Date

ot. No. Page No.

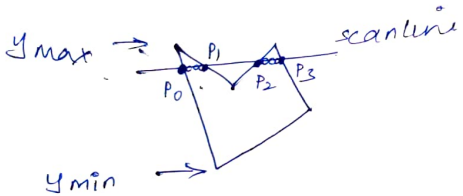
```
    else draw_triangle(a,b,c);
}
void tetrahedron()
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glColor3f(1.0,0,0);
  divide_triangle(tetra[0], tetra[1], tetra[2], m);
  glColor3f(0, 1.0, 0);
  divide_triangle(tetra[3], tetra[2], tetra[1], m);
  glColor3f(0, 0, 1.0);
  divide_triangle(tetra[0], tetra[3], tetra[1], m);
  glColor3f(0,0,0);
  divide_triangle(tetra[0], tetra[2], tetra[3], m);
  glFlush();
}
void myinit4()
{ glClearColor(1, 1, 1.0, 1.0);
  glPointSize(5.0);
  gluOrtho(-500, 500, -500, 500, -500, 500);
}
void p4_main(int argcp, char **argv)
{ cout << "Enter m:"; cin >> m;
  glutInit(&argcp, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_DEPTH);
  glutInitWindowSize(200,300);
  glutInitWindowPosition(100, 100);
  glutCreateWindow("Display");
  glutDisplayFunc(tetrahedron);
  glEnable(GL_DEPTH_TEST); myinit4();
  glutMainLoop();
}
```

Teacher's Signature

Logic

Algorithm for scanline area filling algorithm

1. For a given polygon, find Ymin and Ymax.
2. Scanlines drawn horizontally intersects with each edge of the polygon from Ymin to Ymax. Name each intersection, or store each intersection point.
3. Sort the intersection points in increasing order of x coordinate.
4. Fill each pair of coordinate inside the polygon and ignore the alternate pairs.



Date 6/11/20

Expt. No. 4 Page No.

Q. Write a program to fill any given polygon using scan-line area filling algorithm

```
#include <stdlib.h>
#include <GL/glut.h>
#include <algorithm>
#include <iostream>
#include <windows.h>
using namespace std;
float x[100], y[100];
int n1, m1, wx = 500, wy = 500;
static float intx[100] = {0};
void draw_line (float x1, float y1, float x2, float y2)
{ Sleep(100);
  glColor3f(1,0,0);
  glBegin (GL_LINES);
  glVertex2f(x1, y1);
  glVertex2f(x2, y2);
  glEnd();
  glFlush();
}
void edgeDetect (float x1, float y1, float x2, float y2, int
  scanline)
{ float temp;
  if (y2<y1)
  { temp = x1; x1 = x2; x2=temp;
    temp = y1; y1 = y2; y2 = temp;
  }
  if (scanline > y1 && scanline <y2)
```

Teacher's Signature

```
#include<stdlib.h>
#include<gl/glut.h>
#include<algorithm>
#include<iostream>
#include<windows.h>

using namespace std;
float x[100], y[100]; //= { 0,0,20,100,100 }, y[] = { 0,100,50,100,0
};

int n1, m1;
int wx = 500, wy = 500;
static float intx[10] = { 0 };

void draw_line(float x1, float y1, float x2, float y2) {
        Sleep(100);
        glColor3f(1, 0, 0);
        glBegin(GL_LINES);
        glVertex2f(x1, y1);
        glVertex2f(x2, y2);
        glEnd();
        glFlush();

}

void edgeDetect(float x1, float y1, float x2, float y2, int scanline)
{

        float temp;
        if (y2 < y1) {

                temp = x1; x1 = x2; x2 = temp;
                temp = y1; y1 = y2; y2 = temp;
        }

        if (scanline > y1 && scanline < y2)
                intx[m1++] = x1 + (scanline - y1) * (x2 - x1) / (y2 -
y1);
}

void scanfill(float x[], float y[]) {
        for (int s1 = 0; s1 <= wy; s1++) {
                m1 = 0;
                for (int i = 0; i < n1; i++) {

                        edgeDetect(x[i], y[i], x[(i + 1) % n1], y[(i +
1) % n1], s1);
                }
                sort(intx, (intx + m1));
                if (m1 >= 2)
                        for (int i = 0; i < m1; i = i + 2)
                                draw_line(intx[i], s1, intx[i + 1],
s1);
```

```
        intx[m1++] = x1 + (scanline - y1) * (x2 - x1) / (y2 - y1);
}
void scanfill(float x[], float y[])
{
    for (int s1 = 0; s1 <= wy; s1++)
    {
        m1 = 0;
        for (int i = 0; i < n1; i++)
            edgeDetect(x[i], y[i], x[(i + 1) % n1], y[(i + 1) % n1], s1);
        sort(intx, intx + m1);
        if (m1 >= 2)
            for (int i = 0; i < m1; i = i + 2)
                draw_line(intx[i], s1, intx[i + 1], s1);
    }
}
void display_filled_polygon()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(2);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < n1; i++) glVertex2f(x[i], y[i]);
    glEnd();
    scanfill(x, y);
}
void scanline_Init()
{
    glClearColor(1, 1, 1, 1);
    glColor3f(0, 0, 1);
    glPointSize(1);
    gluOrtho2D(0, wx, 0, wy);
}
void scanline_main(int argc, char** argv)
{
    glutInit(&argc, argv);
```

Date

ot. No.

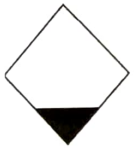
Page No.

Teacher's Signature

C:\Users\saadh\source\repos\opengl\Debug\opengl.exe

```
Enter no. of sides:
4
Enter coordinates of endpoints:
X-coord Y-coord:
100 50
X-coord Y-coord:
150 100
X-coord Y-coord:
100 150
X-coord Y-coord:
50 100
```

scanline



Locals Watch 1

Date

t. No. Page No.

```
printf("Enter no. of sides :\n");
scanf_s("%d",&n);
printf("Enter coordinates of endpoints:\n");
for(int i=0;i<n;i++)
{ printf("x-coord Y-coord:\n");
  scanf_s("%f %f",&x[i],&y[i]);
}
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(300,150);
glutCreateWindow("scanline");
glutDisplayFunc(display_filled_polygon);
scanline_Init();
glutMainLoop();
}
```

Logic

Rotation

If P is the point to be rotated and R is the rotation matrix, then, the point after rotation is $P' = R \cdot P$

$$R(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For rotation about a fixed point,

1. Translate object to origin
2. Rotate the object about the origin
3. Translate the object to its original position from origin. This is reverse translation, i.e.,

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{bmatrix}$$

Where translation for a point P with T as rotation matrix is $P' = T \cdot P$ where

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Date 6/11/20

Expt. No. 5 Page No.

Q. Write a program to create a house like figure and perform the following operations : (i) Rotate it about a given fixed point using OpenGL transformation functions. (ii) Reflect it about an axis $y = mx + c$ using OpenGL transformation functions.

```
#include <GL/glut.h>
#include <math.h>
#include<stdio.h>
float house[11][2] = {{100,200},{200,250},{300,200},
    {100,200},{100,100},{175,100},{145,150},{225,150},
{225,100},{300,100},{300,200}};
int angle;
float m, c, theta;
void display()
{ glClearColor(1,1,1,0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-450,450,-450,450);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glColor3f(1,0,0);
  glBegin(GL_LINE_LOOP);
  for(int i=0; i<11; i++) glVertex2fv(house[i]);
  glEnd();
  glFlush();
  glPushMatrix();
  glTranslatef(100,100,0);
```

Teacher's Signature

Reflection about line $y = mx + c$

1. Slope = m and y intercept = c
2. $m = \tan\theta \Rightarrow \theta = \tan^{-1}(m)$, where θ is the inclination of the line with respect to x-axis.
3. Translate the line $y = mx + c$ to origin, which is

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

4. Rotate the line to coincide with positive x axis, that is in clockwise direction, using θ.
5. Now reflection is performed about x axis given as $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ which is scaling with $S_x = 1, S_y = -1$
6. Now rotate the in anticlockwise direction with $-\theta$, to match the line as got in STEP 3 [through origin].
7. Finally translate $-c$ to obtain original line and resultant image.

We get $T_R = T^{-1} \cdot R^{-1} \cdot M \cdot R \cdot T$

Date

Expt. No.

Page No.

```
glRotatef(angle,0,0,1);
glTranslatef(-100,-100,0);
glColor3f(1,1,0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++) glVertex2fv(house[i]);
glEnd();
glPopMatrix();
glFlush();
}
void display2()
{ glClearColor(1,1,1,0);
  glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-450,450,-450,450);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glColor3f(1,0,0);
  glBegin(GL_LINE_LOOP);
  for(int i=0; i<11; i++) glVertex2fv(house[i]);
  glEnd();
  glFlush();
  float x1=0, x2=500; float y1=m*x1+c; float y2=
  m*x2+c;
  glColor3f(1,1,0);
  glBegin(GL_LINES);
  glVertex2f(x1,y1);
  glVertex2f(x2,y2);
  glEnd(); glFlush();
```

Teacher's Signature

```
#include<gl/glut.h>
#include <math.h>
//#include<stdlib.h>
#include<stdio.h>


//RIGHT CLICK TO SHOW REFLECTED HOUSE


float house[11][2] = { { 100,200 },{ 200,250 },{ 300,200 },{ 100,200
},{ 100,100 },{ 175,100 },{ 175,150 },{ 225,150 },{ 225,100 },{
300,100 },{ 300,200 } };
int angle;
float m, c, theta;
void display5()
{
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-450, 450, -450, 450);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        //NORMAL HOUSE
        glColor3f(1, 0, 0);
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 11; i++)
                glVertex2fv(house[i]);
        glEnd();
        glFlush();
        //ROTATED HOUSE
        glPushMatrix();
        glTranslatef(100, 100, 0);
        glRotatef(angle, 0, 0, 1);
        glTranslatef(-100, -100, 0);
        glColor3f(1, 1, 0);
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 11; i++)
                glVertex2fv(house[i]);
        glEnd();
        glPopMatrix();
        glFlush();
}
void display2()
{
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-450, 450, -450, 450);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        //normal house
        glColor3f(1, 0, 0);
```

Date _____

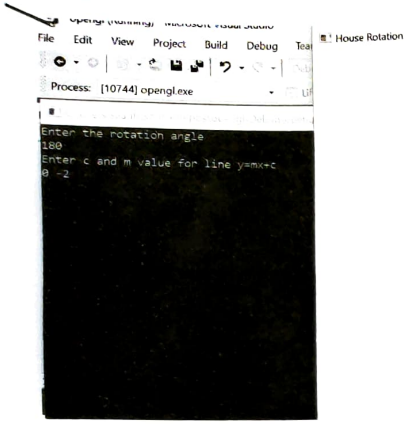
Pt. No. _____ Page No. _____

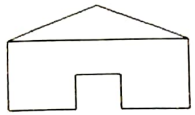
```
glPushMatrix();
glTranslatef(0, c, 0);
theta = atan(m);
theta = theta * 180 / 3.14;
glRotatef(theta, 0, 0, 1);
glScalef(1, -1, 1);
glRotatef(-theta, 0, 0, 1);
glTranslatef(0, -c, 0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++) glVertex2fv(house[i]);
glEnd();
glPopMatrix();
glFlush();
}
glvoid myInit5()
{ glClearColor(1.0, 1.0, 1.0, 1.0);
  glColor3f(1.0, 0.0, 0.0);
  glLineWidth(2.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-450, 450, -450, 450);
}
void mouse(int btn, int state, int x, int y)
{ if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
     display5();
else if (btn == GLUT_RIGHT_BUTTON && state==GLUT_
   DOWN) display2();
}
```

Teacher's Signature _____

12/18/2020

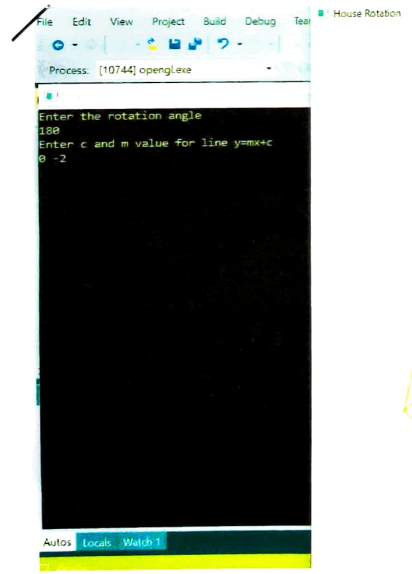
op5_rot.png





12/18/2020

op5_ref.png



Date _____

No. _____ Page No. _____

```
void ps_main(int argc, char** argv)
{
printf("Enter the rotation angle\n");
scanf_s("%d", &angle);
printf("Enter c and m value for line y=mx+c\n");
scanf_s("%f %f", &c, &m);
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(900,900);
glutInitWindowPosition(100,100);
glutCreateWindow("House Rotation");
glutDisplayFunc(display5);
glutMouseFunc(mouse);
myinit5();
glutMainLoop();
}
```

Teacher's Signature _____

Date 6/11/20

Expt. No. 6 Page No.

Logic

Algorithm for Cohen Sutherland line clipping

1. Calculate positions of both endpoints of line
2. Perform OR operation on both of these end points
3. If the OR operation gives 0000,
   line is visible,
   else,
   perform AND operation on both endpoints.
   If AND is not equal to 0000,
   line is invisible
   else,
   line is partially inside and we need to clip it.
4. To perform clipping, find intersection with boundaries of the window with $m = (y2 - y1)/(x2 - x1)$

a) if bit 1 is '1', line intersects with left boundary of the rectangular window.

$y3 = y1 + m(x - x1)$, where $x = x_{wmin}$ (window)

b) If bit 2 is '1', line intersects with right boundary

$y3 = y1 + m(x - x1)$, where $x = x_{wmax}$

c) if bit 3 is '1', line intersects with bottom boundary

$x3 = x1 + (y - y1)/m$, where $y = y_{wmin}$

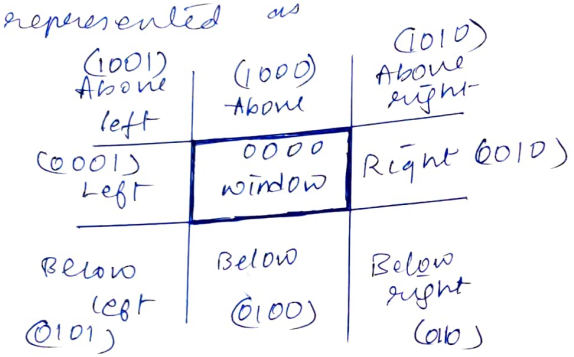
Q. Write a program to implement the cohen-Sutherland line clipping algorithm. Make a provision to specify the input for multiple lines, window for clipping and viewport for displaying the clipped image.

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#define outcode int
#define true 1
#define false 0
double xmin, ymin, xmax, ymax, xvmin, yvmin, xvmax, yvmax;
const int RIGHT = 2;
const int LEFT = 1;
const int TOP = 8;
const int BOTTOM = 4;
int n;
struct line_segment{
 int x1; int y1; int x2; int y2; };
struct line_segment ls[10];
outcode computeoutcode (double x, double y)
{ outcode code = 0;
   if (y > ymax) code |= TOP;
   else if (y < ymin) code |= BOTTOM;
   if (x > xmax) code |= RIGHT;
   else if (x < xmin) code |= LEFT;
  return code;
}
```

Teacher's Signature

d) If bit 4 is '1', line intersects the Top boundary
$x_s = x_1 + (y - y_1)/m$, where $y = y_{wmax}$.

The window and regions surrounding it can be represented as



Date _____

Expt. No. _____ Page No. _____

```
void cohensuther(double x0, double y0, double x1, double y1)
{ outcode outcode0, outcode1, outcodeout;
  bool accept = false, done = false;
  outcode0 = computeoutcode(x0, y0);
  outcode1 = computeoutcode(x1, y1);
  do{
     if (!(outcode0| outcode1))
     { accept = true; done = true; }
     else if (outcode0 & outcode1) done = true;
     else
     { double x, y;
       outcodeout = outcode0? outcode0: outcode1;
       if (outcodeout & TOP)
       { x = x0 + (x1-x0) * (ymax - y0) / (y1-y0);
         y = ymax; }
       else if (outcodeout & BOTTOM)
       { x = x0 + (x1-x0) * (ymin - y0)/(y1-y0);
         y = ymin; }
       else if (outcodeout & RIGHT)
       { y = y0 + (y1-y0) * (xmax - x0) / (x1-x0);
         x = xmax; }
       else
       { y = y0 + (y1-y0) * (xmin - x0) / (x1-x0);
         x = xmin; }
       if (outcodeout == outcode0)
       { x0 = x; y0 = y; outcode0 = computeoutcode(x0,y0); }
       else
       { x1 = x; y1 = y; outcode1 = computeoutcode(x1, y1); }
     }
  } while (!done);
```

Teacher's Signature _____

pi-1

```
#include<stdio.h>
#include<stdlib.h>
#include<gl/glut.h>

#define outcode int
#define true 1
#define false 0
double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;


const int RIGHT = 2;
const int LEFT = 1;
const int TOP = 8;
const int BOTTOM = 4;

int n;
struct line_segment {
        int x1;
        int y1;
        int x2;
        int y2;
};
struct line_segment ls[10];

outcode computeoutcode(double x, double y)
{
        outcode code = 0;
        if (y > ymax)
                code |= TOP;
        else if (y < ymin)
                code |= BOTTOM;
        if (x > xmax)
                code |= RIGHT;
        else if (x < xmin)
                code |= LEFT;

        return code;
}

void cohensuther(double x0, double y0, double x1, double y1)
{
        outcode outcode0, outcode1, outcodeout;
        bool accept = false, done = false;
        outcode0 = computeoutcode(x0, y0);
        //printf("outcode0");
        //printf("%d",outcode0);
        outcode1 = computeoutcode(x1, y1);
        //printf("outcode1");
        //printf("%d", outcode1);
        do
        {
                if (!(outcode0 | outcode1))
                {
```

Date

No. _____ Page No. _____

```
if (accept)
{ double sx = (xvmax - xvmin) / (xmax - xmin);
  double sy = (yvmax - yvmin) / (ymax - ymin);
  double vx0 = xvmin + (x0 - xmin) * sx;
  double vy0 = yvmin + (y0 - ymin) * sy;
  double vx1 = xvmin + (x1 - xmin) * sx;
  double vy1 = yvmin + (y1 - ymin) * sy;
  glColor3f(1,0,0);
  glBegin(GL_LINE_LOOP);
  glVertex2f(xvmin, yvmin);
  glVertex2f(xvmax, yvmin);
  glVertex2f(xvmax, yvmax);
  glVertex2f(xvmin, yvmax);
  glEnd();
  glColor3f(0,0,1);
  glBegin(GL_LINES);
  glVertex2d(vx0, vy0);
  glVertex2d(vx1, vy1);
  glEnd();
}

void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(0,0,1);
  glBegin(GL_LINE_LOOP);
  glVertex2f(xmin, ymin);
  glVertex2f(xmax, ymin);
  glVertex2f(xmax, ymax);
  glVertex2f(xmin, ymax);
```

Teacher's Signature _____

oo6.png

```
Enter window coordinates (xmin ymin xmax ymax):
20 20 100 100
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
150 150 230 230
Enter no. of lines:
2
Enter line endpoints (x1 y1 x2 y2):
10 10 40 40
Enter line endpoints (x1 y1 x2 y2):
0 75 150 75
```

```
glEnd();
for (int i = 0; i<n ; i++)
{   glBegin (GL_LINES);
    glVertex2d (ls[i].x1, ls[i].y1);
    glVertex2d (ls[i].x2, ls[i].y2);
    glEnd();
}
for (int i = 0; i<n; i++)
    cohensutherland (ls[i], x1, ls[i].y1, ls[i].x2, ls[i].y2);
glflush();

id myinit6()
glclearcolor (1,1,1,1); glcolor3f (1,0,0); glPointSize(4.0);
glMatrixMode (GL_PROJECTION); glLoadIdentity();
gluOrtho2D (0, 500, 0, 500);

void p6_main (int argc, char **argv)
{ printf ("Enter window coordinates (xmin ymin xmax ymax):\n");
  scanf_s ("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
printf ("Enter viewport coordinates (xvmin yvmin xvmax yvmax):\n");
scanf_s ("%lf %lf %lf %lf", &xvmin, &yvmin, &xvmax, &yvmax);
printf ("Enter no. of lines:\n"); scanf_s ("%d", &n);
for(int i=0; i<n; i++){ printf ("Enter line endpoints:\n");
scanf_s ("%d %d %d %d", &ls[i].x1, &ls[i].y1, &ls[i].x2, &ls[i].y2);
glutInit (&argc, argv); glutInitDisplayMode (GLUT_SINGLE |
GLUT_RGB); glutInitWindowSize (500, 500);
glutInitWindowPosition (0, 0); glutCreateWindow ("clip");
myinit6(); glutDisplayFunc (display6); glutMainLoop();
}
```

Date _____

Page No. _____

Teacher's Signature _____

Logic

Algorithm for Liang-Barsky line clipping

1. Set tmin = 0, tmax = 1
2. Calculate the values of t for t(left), t(right), t(top), t(bottom)
   i. If t < tmin, ignore and move to the next edge
   ii. else, seperate the t values as entering or exiting values using the inner product
   iii. if t is entering value, set tmin = t, and if t is exciting value, set tmax = t
3. If tmin < tmax, draw a line from (x1 + tmin(x2 - x1), y1 + tmin(y2 - y1)) to (x1 + tmax(x2 - x1), y1 + tmax(y2 - y1))
4. If the line crosses the window, (x1 + tmin(x2 - x1), y1 + tmin(y2 - y1)) and (x1 + tmax(x2 - x1), y1 + tmax(y2 - y1)) are the intersection point of line and edge.

Date 20/11/20

Expt. No. 7

Page No.

8. Write a program to implement the Liang-Barsky line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping and viewport for displaying the clipped image.

```
#include<stdio.h>
#include<GL/glut.h>
double xmin=7, ymin=7, xmax=7, ymax=7, xvmin=7, yvmin=7, xvmax=7, yvmax=7;
int n=7;
struct line_segment{
    int x1; int y1; int x2; int y2; }; struct line_segment ls[10];
int cliptest(double p, double q, double* u1, double* u2)
{ double r;
  if(p) r=q/p;
  if(p<0.0)
  { if(r>*u1) *u1=r;
    if(r>*u2) return(false); }
  else
    if(p>0.0)
    { if(r<*u2) *u2=r;
      if(r<*u1) return(false); }
  else
    if(p==0.0) { if(q<0.0) return(false); }
  return(true);
}
void LiangBarskylineClipAndDraw(double x0, double y0,
    double x1, double y1)
{ double dx=x1-x0, dy=y1-y0, u1=0.0, u2=1.0;
  glColor3f(1.0, 0.0, 0.0);
  glBegin(GL_LINE_LOOP);
```

Teacher's Signature

```
#include <stdio.h>
#include <GL/glut.h>

double xmin7, ymin7, xmax7, ymax7; //50 50 100 100
double xvmin7, yvmin7, xvmax7, yvmax7; //200 200 300 300

int n7;

struct line_segment7 {
        int x1;
        int y1;
        int x2;
        int y2;
};
struct line_segment7 ls[10];
int cliptest(double p, double q, double* u1, double* u2)
{
        double r;
        if (p) r = q / p;  // to check whether p
        if (p < 0.0)    // potentially entry point, update te
        {
                if (r > * u1)* u1 = r;
                if (r > * u2) return(false); // line portion is
 outside
        }
        else
                if (p > 0.0)    //  Potentially leaving point, update
 tl
                {
                        if (r < *u2)* u2 = r;
                        if (r < *u1) return(false); // line portion is
 outside
                }
                else
                        if (p == 0.0)
                        {
                                if (q < 0.0) return(false); // line
 parallel to edge but outside
                        }
        return(true);
}

void LiangBarskyLineClipAndDraw(double x0, double y0, double x1,
double y1)
{
        double dx = x1 - x0, dy = y1 - y0, u1 = 0.0, u2 = 1.0;
        //draw a red colored viewport
        glColor3f(1.0, 0.0, 0.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xvmin7, yvmin7);
        glVertex2f(xvmax7, yvmin7);
        glVertex2f(xvmax7, yvmax7);
        glVertex2f(xvmin7, yvmax7);
        glEnd();
```

Date

o.

Page No.

```
glVertex2f(xvmin7, yvmin7);
glVertex2f(xvmax7, yvmin7);
glVertex2f(xvmax7, yvmax7);
glVertex2f(xvmin7, yvmax7);
glEnd();
if (cliptest(-dx, x0-xmin7, &u1, &u2))
 if (cliptest(dx, xmax7-x0, &u1, &u2))
  if (cliptest(-dy, y0-ymin7, &u1, &u2))
   if (cliptest(dy, ymax7-y0, &u1, &u2))
    { if (u2 < 1.0)
      { x1 = x0+u2*dx; y1 = y0+u2*dy; }
      if (u2 > 1.0)
      { x0 = x0+u1*dx; y0 = y0+u1*dy; }
double sx = (xvmax7-xvmin7) / (xmax7-xmin7);
double sy = (yvmax7-yvmin7) / (ymax7-ymin7);
double vx0 = xvmin7 + (x0-xmin7) *sx;
double vy0 = yvmin7 + (y0-ymin7) *sy;
double vx1 = xvmin7 + (x1-xmin7) *sx;
double vy1 = yvmin7 + (y1-ymin7) * sy;
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINES);
glVertex2d(vx0, vy0);
glVertex2d(vx1, vy1);
glEnd();
}
}
void display()
{ glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0, 0.0, 0.0);
  for(int i=0; i<n7; i++){
```

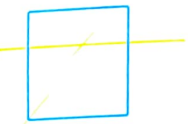
Teacher's Signature

op7.png

12/18/2020

Liang Barsky Line Clipping Algorithm

```
Enter window coordinates: (xmin ymin xmax ymax)
30 30 120 120
Enter viewport coordinates: (xvmin yvmin xvmax yvmax)
170 170 250 250
Enter no. of lines:
3
Enter coordinates: (x1 y1 x2 y2)
5 5 50 50
Enter coordinates: (x1 y1 x2 y2)
0 90 170 90
Enter coordinates: (x1 y1 x2 y2)
70 80 90 100
```



Date

Expt. No.

Page No.

```
glBegin (GL_LINES);
glVertex2D (ls[i].x1, ls[i].y1);
glVertex2D (ls[i].x2, ls[i].y2);
glEnd();
}
glColor3f (0.0, 0.0, 1.0);
glBegin (GL_LINE_LOOP);
glVertex2f (xmin7, ymin7);
glVertex2f (xmax7, ymin7);
glVertex2f (xmax7, ymax7);
glVertex2f (xmin7, ymax7);
glEnd();
for (int i=0; i<n7; i++) LiangBarskyLineClipAndDraw(ls
  ls[i].y1, ls[i].x2, ls[i].y2);
glFlush();
}
void myinit7()
{ glClearColor(1.0,1.0,1.0,1.0); glColor3f(1.0, 0.0, 0.0);
  glLineWidth(2.0); glMatrixMode(GL_PROJECTION)
  glLoadIdentity(); gluOrtho2D(0.0, 499.0, 0.0, 499.
void p7_main (int argc, char *argv)
{ glutInit (&argc, argv); glutInitDisplayMode(GLUT_
  GLUT_RGB); glutInitWindowSize(500,500);
  glutInitWindowPosition(0,0);
  printf("Enter window coordinates\n"); scanf_s("%
  %lf %lf", &xmin7, &ymin7, &xmax7, &ymax7);
  printf("Enter viewport coordinates\n"); scanf_s("%
  %lf, %lf, %lf", &xvmin7, yvmin7, xvmax7, yvmax
  printf("Enter no of lines\n"); scanf_s("%d", &n
```

Teacher's Signature

Date _____

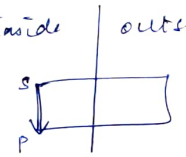
Expt. No. _____ Page No. _____

```
for(int i=0; i<n*2; i++)
{ printf("Enter coordinates"); scanf_s("%d %d %d %d",
&ls[i].x1, &ls[i].y1, &ls[i].x2, &ls[i].y2);
}
glutCreateWindow("Liang Barsky line clipping
    algorithm");
glutDisplayFunc(display2);
myinit2();
glutMainLoop();
}
```

Logic

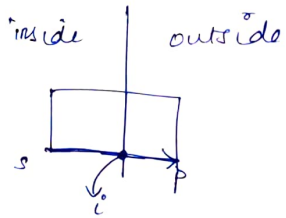
1. The vertices of the polygon are clipped against each edge of the clipping window.
2. First, the polygon is clipped against the left edge of the polygon window and so on, that is, it uses divide and conquer strategy
3. We encounter a main consider the current vertex position be p and previous vertex have position s. Then,

Case (i) both s and p inside the clipping window edge.



Here, we save p.

Case (ii) s inside, but p outside the clipping window edge



If i is the point of intersection, save i.

Date 20/11/20

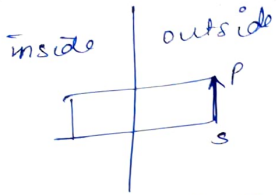
Expt. No. 8 Page No.

Q. Write a program to implement the Cohen-Hodgeman polygon clipping algorithm. Make a provision to specify the input polygon and window for clipping.

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
int poly_size, poly_points[20][2], org_poly_size, org_poly_points[20][2],
clipper_size, clipper_points[20][2];
const int MAX_POINTS = 20;
void drawPoly( int p[][2], int n)
{ glBegin (GL_POLYGON);
  for (int i=0; i<n; i++)
    glVertex2f( p[i][0], p[i][1]);
  glEnd(); }
int x_intersect (int x1, int y1, int x2, int y2, int x3, int y3,
    int x4, int y4)
{ int num = (x1 * y2 - y1 * x2) * (x3-x4) - (x1-x2) *
        (x3 * y4 - y3 * x4);
  int den = (x1 - x2) * (y3-y4) - (y1-y2) * (x3-x4);
  return num/den; }
int y_intersect( int x1, int y1, int x2, int y2, int x3,
  int y3, int x4, int y4)
{ int num = (x1 * y2 - y1 * x2) * (y3-y4) - (y1-y2) *
        (x3 * y4 - y3 * x4);
  int den = (x1-x2) * (y3-y4) - (y1-y2) * (x3-x4);
  return num/den;
}
```

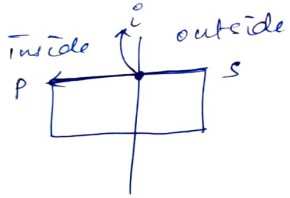
Teacher's Signature

Case (iii) both s and p outside the clipping window edge.



Here, we do not save any point/ vertex.

Case (iv) p inside, but s outside the clipping window edge



let i be the point of intersection.

Here, we save i and p vertices.

4. The newly found points are used to form the resulting polygon after clipping.

Date

Expt. No.

Page No.

```
void clip(int poly_points[][2], int& poly_size, int x1, int y1,
  int x2, int y2)
{ int new_points[MAX_POINTS][2], new_poly_size = 0;
  for (int i = 0; i < poly_size; i++)
  { int k = (i+1) % poly_size;
    int ix = poly_points[i][0], iy = poly_points[i][1];
    int kx = poly_points[k][0], ky = poly_points[k][1];
    int i_pos = (x2-x1) * (iy-y1) - (y2-y1) * (ix-x1);
    int k_pos = (x2-x1) * (ky-y1) - (y2-y1) * (kx-x1);
    if (i_pos >= 0 && k_pos >= 0)
    { new_points[new_poly_size][0] = kx;
      new_points[new_poly_size][1] = ky;
      new_poly_size++;
    } else if (i_pos < 0 && k_pos >= 0)
    { new_points[new_poly_size][0] = x_intersect(x1,
        y1, x2, y2, ix, iy, kx, ky);
      new_points[new_poly_size][1] = y_intersect(x1,
        y1, x2, y2, ix, iy, kx, ky);
      new_poly_size++;
      new_points[new_poly_size][0] = kx;
      new_points[new_poly_size][1] = ky;
      new_poly_size++;} else if (i_pos >= 0 && k_
    { new_points[new_poly_size][0] = x_intersec
      (x1, y1, x2, y2, ix, iy, kx, ky);
      new_points[new_poly_size][1] = y_inters
      (x1, y1, x2, y2, ix, iy, kx, ky);
      new_poly_size++} else { }
  }
  poly_size = new_poly_size;
```

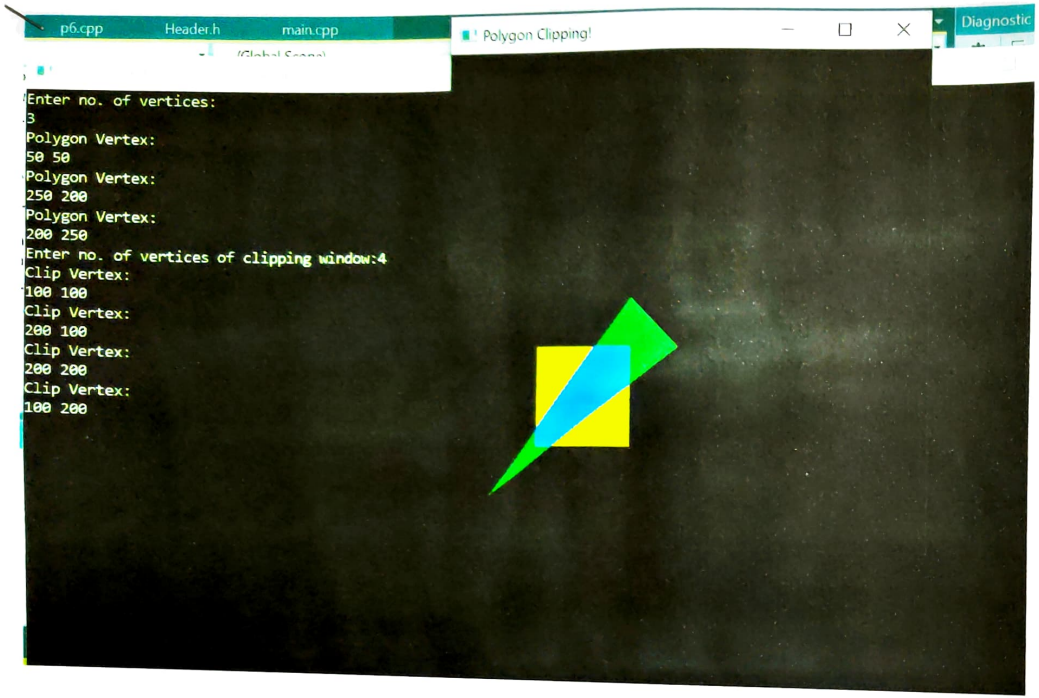
Teacher's Signature

```
// C++ program for implementing Sutherland–Hodgman
// algorithm for polygon clipping
#include<iostream>
#include<GL/glut.h>
using namespace std;
int poly_size, poly_points[20][2], org_poly_size, org_poly_points[20]
[2], clipper_size, clipper_points[20][2];
const int MAX_POINTS = 20;


// Returns x-value of point of intersection of two
// lines

void drawPoly(int p[][2], int n) {
        glBegin(GL_POLYGON);
        for (int i = 0; i < n; i++)
                glVertex2f(p[i][0], p[i][1]);
        glEnd();
}

int x_intersect(int x1, int y1, int x2, int y2,
        int x3, int y3, int x4, int y4)
{
        int num = (x1 * y2 - y1 * x2) * (x3 - x4) -
                (x1 - x2) * (x3 * y4 - y3 * x4);
        int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
        return num / den;
}

// Returns y-value of point of intersectipn of
// two lines
int y_intersect(int x1, int y1, int x2, int y2,
        int x3, int y3, int x4, int y4)
{
        int num = (x1 * y2 - y1 * x2) * (y3 - y4) -
                (y1 - y2) * (x3 * y4 - y3 * x4);
        int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
        return num / den;
}

// This functions clips all the edges w.r.t one clip
// edge of clipping area
void clip(int poly_points[][2], int& poly_size,
        int x1, int y1, int x2, int y2)
{
        int new_points[MAX_POINTS][2], new_poly_size = 0;

        // (ix,iy),(kx,ky) are the co-ordinate values of
        // the points
        for (int i = 0; i < poly_size; i++)
        {
                // i and k form a line in polygon
                int k = (i + 1) % poly_size;
                int ix = poly_points[i][0],
```

Expt. No. _____ Date _____ Page No. _____

```
for (int i = 0; i < poly_size; i++)
{ poly_points[i][0] = new_points[i][0];
  poly_point[i][1] = new_points[i][1]; }
}
void init8()
{ glClearColor(0.0f, 0.0f, 0.0f, 0.0f); glMatrixMode(GL_PROJECTION);
  glLoadIdentity(); glOrtho(0.0, 500.0, 0.0, 500.0, 0.0, 500.0);
  glClear(GL_COLOR_BUFFER_BIT); }
void display8()
{ init8();
  glColor3f(1.0f, 0.0f, 0.0f);
  drawPoly(clipper_points, clipper_size);
  glColor3f(0.0f, 1.0f, 0.0f);
  drawPoly(org_poly_points, org_poly_size);
  for(int i=0; i<clipper_size; i++)
  { int k=(i+1)%clipper_size;
    clip(poly_points, poly_size, clipper_points[i][0],
    clipper_points[i][1], clipper_points[k][0], clipper_points[k][1]);
  } glColor3f(0.0f, 0.0f, 1.0f); drawPoly(poly_points, poly_size); glFlush();
}
void pe_main(int argc, char **argv)
{ printf("Enter no of vertices:\n"); scanf("%d", &poly_size);
  org_poly_size = poly_size;
  for(int i=0; i<poly_size; i++)
  { printf("Polygon vertex:\n"); scanf("%d%d", &poly_points
    [i][0], &poly_points[i][1]);
    org_poly_points[i][0] = poly_points[i][0];
    org_poly_points[i][1] = poly_points[i][1]; }
```

Teacher's Signature _____



Date

Expt. No. Page No.

```
printf("Enter no. of vertices of clipping window:");
scanf_s("%d", &clipper_size);
for(int i=0; i<clipper_size; i++)
{ printf("Clip vertex:\n");
  scanf_s("%d %d", &clipper_points[i][0], &clipper_points[i
}

glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(400, 400);
glutInitWindowPosition(100, 100);
glutCreateWindow("Polygon Clipping!");
glutDisplayFunc(display);
glutMainLoop();
}
```

```
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
#define CAR 1
#define WHEEL 2
float s = 1;
void carlist() {

        glNewList(CAR, GL_COMPILE);
        glColor3f(1, 1, 1);
        glBegin(GL_POLYGON);
        glVertex3f(0, 25, 0);
        glVertex3f(90, 25, 0);
        glVertex3f(90, 55, 0);
        glVertex3f(80, 55, 0);
        glVertex3f(20, 75, 0);
        glVertex3f(0, 55, 0);
        glEnd();
        glEndList();

}
void wheellist() {
        glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
        glColor3f(0, 1, 1);
        glutSolidSphere(10, 25, 25);
        glEndList();
}
void mykeyboard(unsigned char key, int x, int y) {
        switch (key) {
        case 't': glutPostRedisplay();
                break;
        case 'q': exit(0);
        default: break;

        }
}

void myInit9() {
        glClearColor(0, 0, 0, 0);
        glOrtho(0, 600, 0, 600, 0, 600);

}
void draw_wheel() {
        glColor3f(0, 1, 1);
        glutSolidSphere(10, 25, 25);

}

void moveCar(float s) {
        glTranslatef(s, 0.0, 0.0);
        glCallList(CAR);
        glPushMatrix();
        glTranslatef(25, 25, 0.0);        //move to first wheel position
        //draw wheel();
```

Date 20/11/20

Expt. No. 9

Page No.

Write a program to model a car like figure using display lists and move a car from one end of the screen to other end. User is able to control the speed with mouse

```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
#define CAR 1
#define WHEEL 2
float s = 1;
void carlist()
{   glNewList(CAR, GL_COMPILE);
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex3f(0, 25, 0);
    glVertex3f(90, 25, 0);
    glVertex3f(90, 55, 0);
    glVertex3f(80, 55, 0);
    glVertex3f(20, 75, 0);
    glVertex3f(0, 55, 0);
    glEnd();
    glEndList();
}

void wheellist()
{   glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
    glColor3f(0, 1, 0);
    glutSolidSphere(10, 25, 25);
    glEndList();
}
```

Teacher's Signature

12/18/2020

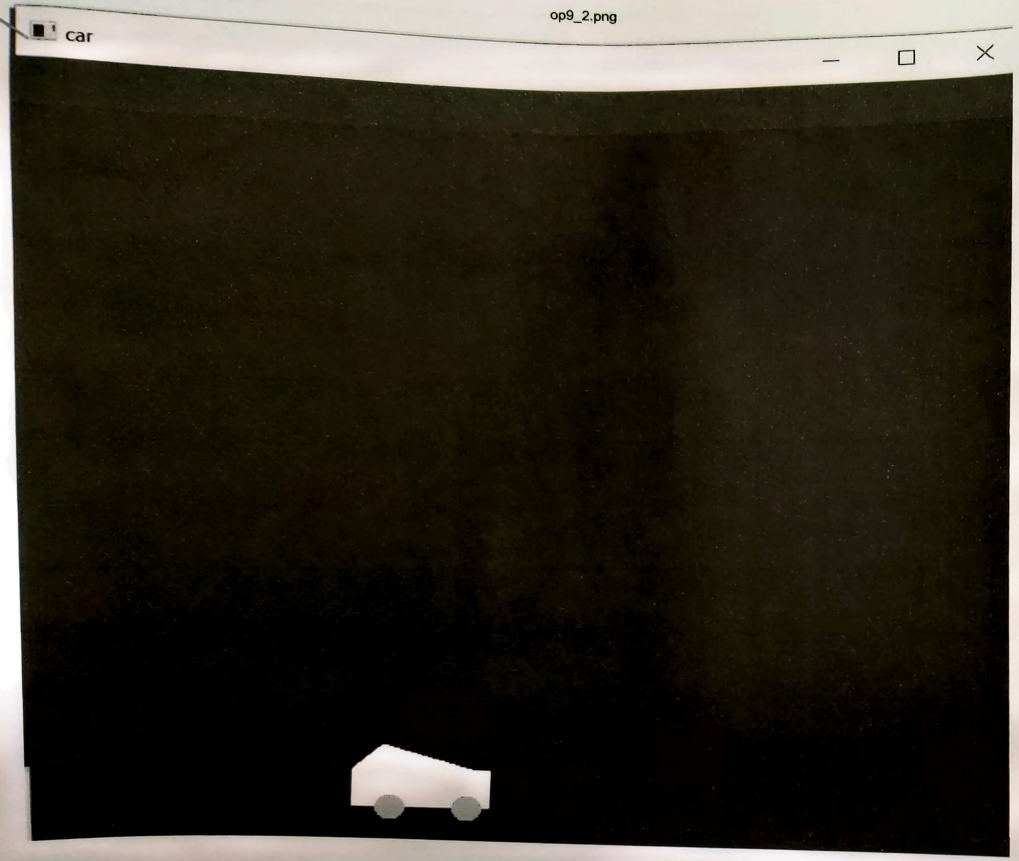
car

Expt. No. _____ Date _____ Page No. _____

```
void mykeyboard(unsigned char key, int x, int y)
{ switch (key)
  { case 't': glutPostRedisplay();
              break;
    case 'q': exit(0);
    default: break;
  }
}
void myinit()
{ glClearColor(0,0,0,0); glOrtho(0,600,0,600,0,600); }
void draw_wheel()
{ glColor3f(0,1,1); glutSolidSphere(10,20,20); }
void moveCar(float s)
{ glTranslatef(s, 0.0, 0.0);
  glCallList(CAR);
  glPushMatrix();
  glTranslatef(25, 25, 0.0);
  glCallList(WHEEL);
  glPopMatrix();
  glPushMatrix();
  glTranslatef(75, 25, 0.0);
  glCallList(WHEEL);
  glPopMatrix();
  glFlush();
}
void myDisp()
{ glClear(GL_COLOR_BUFFER_BIT);
  carlist();
  moveCar(s);
  wheellist();
}
```

Teacher's Signature _____

12/18/2020



Date

Page No.

xpt. No.

```
void mouseq(int btn, int state, int x, int y)
{  if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
     { st=5; myDisp9();}
   else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
     { st=2; myDisp9();}
}
void p9_main (int argc, char **argv)
{ glutInit(&argc, argv);
  glutInitDisplayMode( GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (600, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow("car");
  myInit9();
  glutDisplayFunc(myDisp9);
  glutMouseFunc(mouseq);
  glutKeyboardFunc(myKeyboard);
  glutMainLoop();
}
```

```
#include <stdlib.h>
#include <GL/glut.h>
#include<gl\GL.h>
#include<gl\GLU.h>
#include <time.h>

GLfloat vertices[] = { -1.0,-1.0,-1.0,1.0,-1.0,-1.0,
1.0,1.0,-1.0, -1.0,1.0,-1.0, -1.0,-1.0,1.0,
1.0,-1.0,1.0, 1.0,1.0,1.0, -1.0,1.0,1.0 };

GLfloat normals[] = { -1.0,-1.0,-1.0,1.0,-1.0,-1.0,
1.0,1.0,-1.0, -1.0,1.0,-1.0, -1.0,-1.0,1.0,
1.0,-1.0,1.0, 1.0,1.0,1.0, -1.0,1.0,1.0 };

GLfloat colors[] = { 0.0,0.0,0.0,1.0,0.0,0.0,
1.0,1.0,0.0, 0.0,1.0,0.0, 0.0,0.0,1.0,
1.0,0.0,1.0, 1.0,1.0,1.0, 0.0,1.0,1.0 };

GLubyte cubeIndices[] = {
0,3,2,1,2,3,7,6,0,4,7,3,1,2,6,5,4,5,6,7,0,1,5,4 };

static GLfloat theta[] = { 0.0,0.0,0.0 };
static GLfloat beta[] = { 0.0,0.0,0.0 };
static GLint axis = 2;

void delay(float secs)
{
        float end = clock() / CLOCKS_PER_SEC + secs;
        while ((clock() / CLOCKS_PER_SEC) < end);
}

void displaySingle(void)
{
        /* display callback, clear frame buffer and z buffer,
           rotate cube and draw, swap buffers */

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glRotatef(theta[0], 1.0, 0.0, 0.0);
        glRotatef(theta[1], 0.0, 1.0, 0.0);
        glRotatef(theta[2], 0.0, 0.0, 1.0);

        glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);

        glBegin(GL_LINES);
        glVertex3f(0.0, 0.0, 0.0);
        glVertex3f(1.0, 1.0, 1.0);
        glEnd();

        glFlush();
}
```

Date 27/11/20

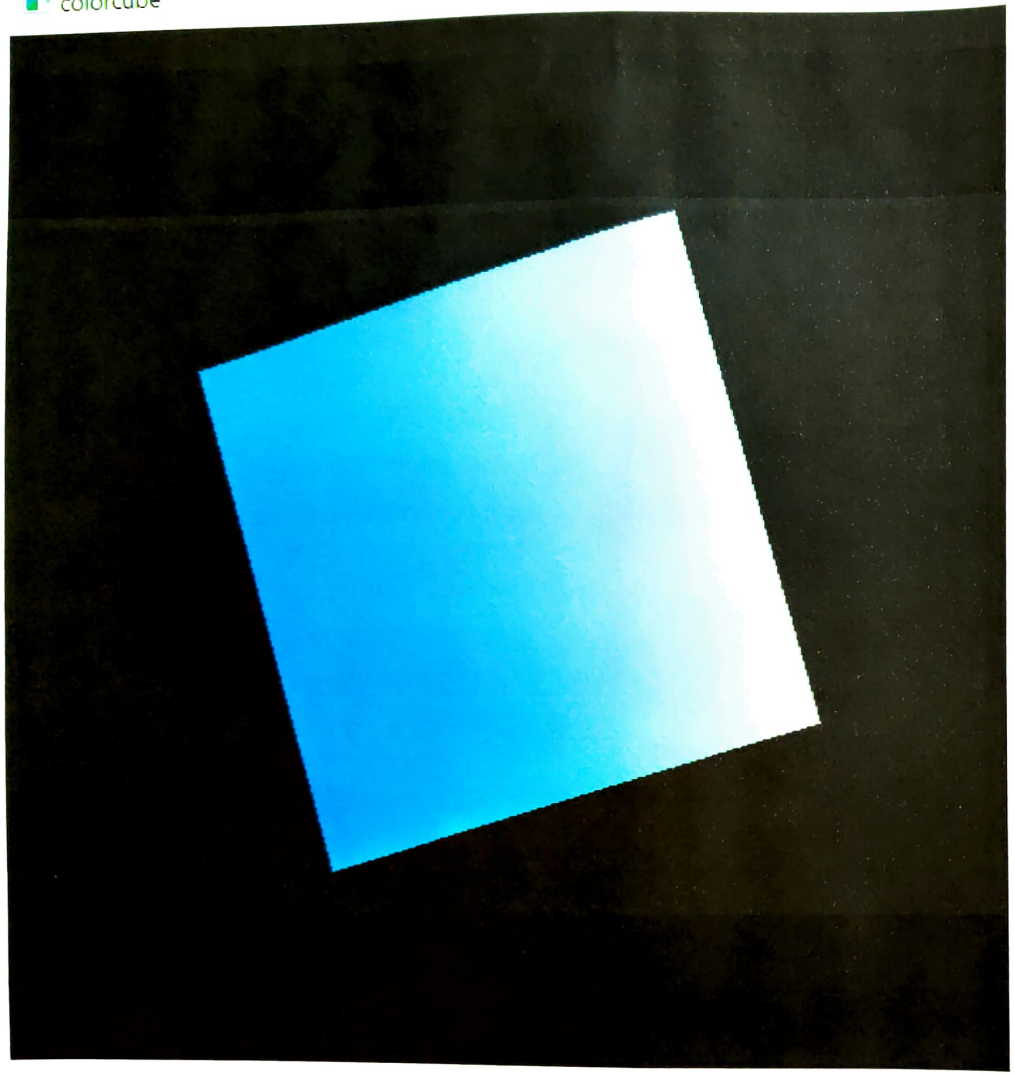
Page No.

Expt. No. 10

. Write a program to create a color cube and spin it using OpenGL transformations.

```
#include <stdlib.h>
#include<GL/glut.h>
#include<time.h>
GLfloat vertices[] ={-1.0,-1.0,-1.0,1.0,-1.0,-1.0,1.0,
1.0,-1.0,-1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,1.0,1.0,
1.0,1.0,-1.0,1.0,1.0};
GLfloat normals[] ={-1.0,-1.0,-1.0,1.0,-1.0,-1.0,1.0,1.0,
-1.0,-1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,1.0,1.0,
1.0,1.0,-1.0,1.0,1.0};
GLfloat colors[] ={0.0,0.0,0.0,1.0,0.0,0.0,1.0,1.0,0.0,
0.0,1.0,0.0,0.0,0.0,1.0,1.0,0.0,1.0,1.0,1.0,1.0,
0.0,1.0,1.0};
GLubyte cubeIndices[] ={0,3,2,1,2,3,7,6,0,4,7,3,
1,2,6,5,4,5,6,7,0,1,5,4};
static GLfloat theta[] ={0.0,0.0,0.0};
static GLfloat beta[] ={0.0,0.0,0.0};
static GLint axis =2;
void delay(float secs)
{ float end = clock()/CLOCKS_PER_SEC +secs;
  while((clock()/(CLOCKS_PER_SEC)<end);
}
void displaySingle(void)
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_
  glLoadIdentity();
  glRotatef(theta[0],1.0,0.0,0.0);
  glRotatef(theta[1],0.0,1.0,0.0);
```

colorcube



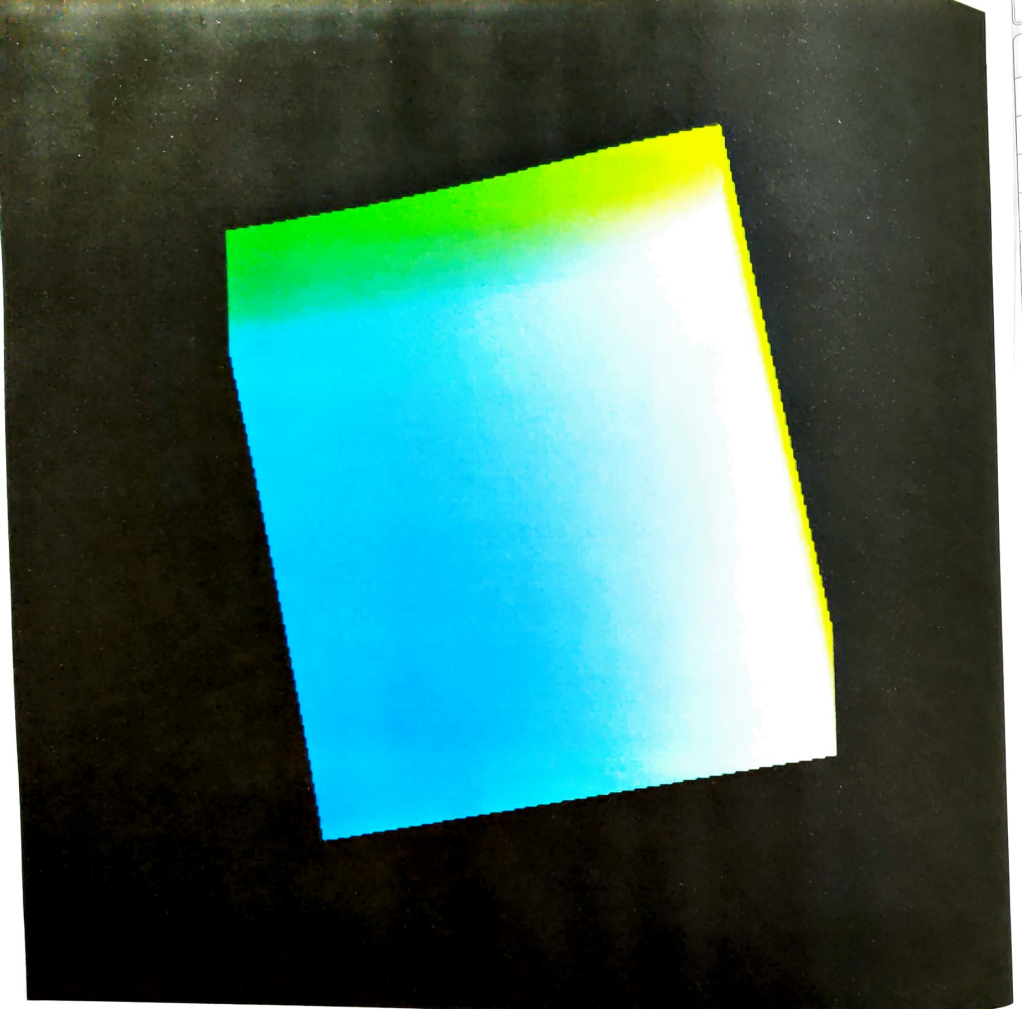
Date

Expt. No.

Page No.

```
glRotatef (Theta[2], 0.0, 0.0, 1.0);
glDrawElements (GL_QUADS, 24, GL_UNSIGNED_BYTE,
    cubeIndices);
glBegin (GL_LINES);
glVertex3f (0.0, 0.0, 0.0);
glVertex3f (1.0, 1.0, 1.0);
glEnd ();
glFlush ();
}

void spinCube ()
{ delay (0.01);
  Theta[axis] += 2.0;
  if ( Theta[axis] > 360.0) Theta[axis] -= 360.0;
  glutPostRedisplay ();
}

void mouse (int btn, int state, int x, int y)
{  if (btn == GLUT_LEFT_BUTTON && state == GLUT_
     DOWN) axis = 0;
   if (btn == GLUT_RIGHT_BUTTON && state == GLUT_
     DOWN) axis = 2;
   if (btn == GLUT_MIDDLE_BUTTON && state ==
     GLUT_DOWN) axis = 1;
}

void myReshape (int w, int h)
{  glViewport (0, 0, w, h);
   glMatrixMode (GL_PROTECTION);
   glLoadIdentity ();
   if (w <= h)
   glOrtho (-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w, 2.0 *
```

Teacher's Signature



```
(GLfloat)h / (GLfloat)w, -10.0, 10.0);
else
  glOrtho(-2.0 * (GLfloat)w/(GLfloat)h, 2.0 * (GLfloat)w/
    (GLfloat)h, -2.0, 2.0, -10.0, 10.0);
  glMatrix (GL_MODELVIEW);
}
void PIO_main(int argc, char **argv)
{ glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(100, 100);
  glutInitWindowSize (500, 500);
  glutCreateWindow("color cube");
  glutReshapeFunc (myReshape);
  glutDisplayFunc (displaySingle);
  glutIdleFunc (spinCube);
  glutMouseFunc (mouse1D);
  glEnable (GL_DEPTH_TEST);
  glEnableClientState (GL_COLOR_ARRAY);
  glEnableClientState (GL_NORMAL_ARRAY);
  glEnableClientState (GL_VERTEX_ARRAY);
  glVertexPointer (3, GL_FLOAT, 0, vertices);
  glColorPointer (3, GL_FLOAT, 0, colors);
  glNormalPointer (GL_FLOAT, 0, normals);
  glColor3f (1.0, 1.0, 1.0);
  glutMainLoop ();
}
```

```
#include<gl/glut.h>
#include<math.h>
#include<stdio.h>
struct screenPt {
        int x;
        int y;
};
typedef enum { limacon = 1, cardioid = 2, threeLeaf = 3, spiral = 4 }
curveName;
int w = 600, h = 500;
int curve = 1;
int red = 0, green = 0, blue = 0;
void myinit11(void) {
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glMatrixMode(GL_PROJECTION);
        gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
void lineSegment(screenPt p1, screenPt p2) {

        glBegin(GL_LINES);
        glVertex2i(p1.x, p1.y);
        glVertex2i(p2.x, p2.y);
        glEnd();
        glFlush();
}
void drawCurve(int curveNum) {
       const double twoPi = 6.283185;
       const int a = 175, b = 60;
       float r, theta, dtheta = 1.0 / float(a);
       int x0 = 200, y0 = 250;
       screenPt curvePt[2];
       curve = curveNum;
       glColor3f(red, green, blue);
       curvePt[0].x = x0;
       curvePt[0].y = y0;
       glClear(GL_COLOR_BUFFER_BIT);
       switch (curveNum) {
       case limacon: curvePt[0].x += a + b; break;
       case cardioid: curvePt[0].x += a + a; break;
       case threeLeaf: curvePt[0].x += a; break;
       case spiral: break;
       default: break;
       }
       theta = dtheta;
       while (theta < twoPi) {
               switch (curveNum) {
               case limacon: r = a * cos(theta) + b; break;
               case cardioid: r = a * (1 + cos(theta)); break;
               case threeLeaf: r = a * cos(3 * theta); break;
               case spiral: r = (a / 4.0) * theta; break;
               default: break;
               }
```

Date 8/11/21

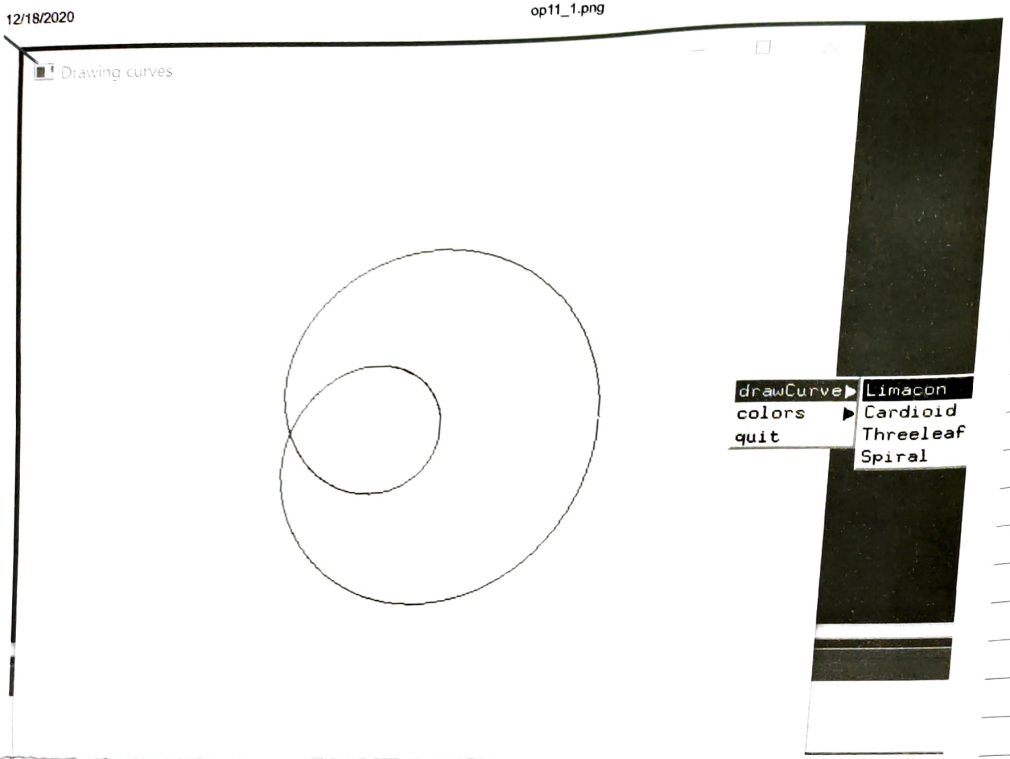
Page No.

Lt. No. 11

Create a menu with three entries named curves, colors and quit. The entry curves has a submenu which has four entries named Limacon, Cardiod, Three-leaf and Spiral. The color menu has submenu with all eight colors of RGB model. Write a program to create the above hierarchical menu and attach appropriate services to each menu entry with mouse buttons.

```
#include<GL/glut.h>
#include<math.h>
#include<stdio.h>
Struct screenPt { int x; int y; };
typedef enum { limacon = 1, cardioid = 2, threeLeaf = 3, spiral =
curveName;
int w = 600, h = 500, curve = 1, red = 0, green = 0, blue = 0;
void myinit11 (void){
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glMatrixMode (GL-PROJECTION);
    glOrtho2D (0.0, 200.0, 0.0, 150.0);
}
void lineSegment (screenPt p1, screenPt p2) {
    glBegin (GL-LINES);
    glVertex2i (p1.x, p1.y);
    glVertex2i (p2.x, p2.y);
    glEnd();
    glFlush();
}
void drawCurve (int curveNum){
    const double twoPi = 6.283185;
    const int a = 175, b = 60;
```

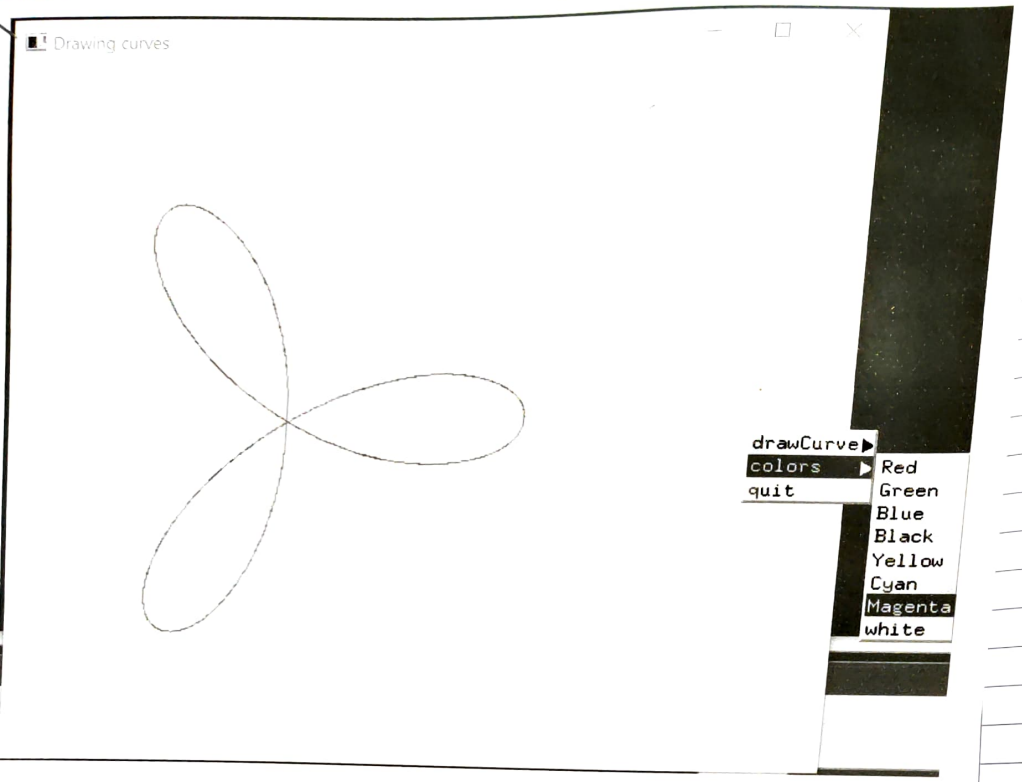
Teacher's Signature



Expt. No. _____ Date _____ Page No. _____

```
float r, theta, dtheta = 1.0/float(a);
int x0 = 200, y0 = 250;
screenPt curvePt[2];
curve = curveNum;
glColor3f(red, green, blue);
curvePt[0].x = x0; curvePt[0].y = y0;
glClear(GL_COLOR_BUFFER_BIT);
switch curveNum) {
  case limacon: curvePt[0].x += a + b; break;
  case cardioid: curvePt[0].x += a + a; break;
  case threeleaf: curvePt[0].x += a; break;
  case spiral: break;
  default: break;
}
theta = dtheta;
while (theta < twoPi) {
    switch (curveNum) {
      case limacon: r = a*cos(theta) + b; break;
      case cardioid: r = a * (1 + cos(theta)); break;
      case threeLeaf: r = a*cos(3*theta); break;
      case spiral: r = a/4.0 * theta; break;
      default; break; }
    curvePt[1].x = x0 + r * cos(theta);
    curvePt[1].y = y0 + r * sin(theta);
    lineSegment(curvePt[0], curvePt[1]);
    curvePt[0].x = curvePt[1].x;
    curvePt[0].y = curvePt[1].y;
    theta += dtheta;
  }
}
```

Teacher's Signature _____



Expt. No. _____ Page No. _____

```
void colorMenu (int id) {
    switch (id) {
        case 0 : break;
        case 1 : red = 0; green = 0; blue = 1; break;
        case 2 : red = 0; green = 1; blue = 0; break;
        case 4 : red = 1; green = 0; blue = 0; break;
        case 8 : red = 0; green = 1; blue = 1; break;
        case 5 : red = 1; green = 0; blue = 1; break;
        case 6 : red = 1; green = 1; blue = 0; break;
        case 7 : red = 1; green = 1; blue = 1; break;
        default : break;
    }
    drawCurve (curve);
}
void main_menu (int id)
{ switch (id) {
        case 3 : exit(0); default : break; }
}
void mydisplay() {}
void myreshape (int nw, int nh)
{ glMatrixMode (GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D (0.0, (double) nw, 0.0, (double) nh);
  glClear (GL_COLOR_BUFFER_BIT);
}
void pll_main (int argc, char * argv[])
{ glutInit (&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize (w, h);
```

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

```
glutInitWindowPosition(100, 100);
glutCreateWindow("Drawing Curve");
int curveId = glutCreateMenu(drawCurve);
glutAddMenuEntry("Limacon", 1);
glutAddMenuEntry("Cardioid", 2);
glutAddMenuEntry("Three leaf", 3);
glutAddMenuEntry("Spiral", 4);
glutAttachMenu(GLUT_LEFT_BUTTON);
int colorId = glutCreateMenu(colorMenu);
glutAddMenuEntry("Red", 4);
glutAddMenuEntry("Green", 2);
glutAddMenuEntry("Blue", 1);
glutAddMenuEntry("Black", 0);
glutAddMenuEntry("Yellow", 6);
glutAddMenuEntry("Cyan", 3);
glutAddMenuEntry("Magenta", 5);
glutAddMenuEntry("White", 7);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawCurve", curveId);
glutAddSubMenu("colors", colorId);
glutAddMenuEntry("quit", 3);
glutAttachMenu(GLUT_LEFT_BUTTON);
myinit();
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);
glutMainLoop();
}
```

Teacher's Signature _____

```
#include<iostream>
#include<math.h>
#include<gl/glut.h>

using namespace std;
float f, g, r, x1[4], yc[4];
int flag = 0;
void myInit12() {

        glClearColor(1, 1, 1, 1);
        glColor3f(1, 1, 1);
        glPointSize(5);
        gluOrtho2D(0, 500, 0, 500);
}

void drawPixel(float x, float y) {
        glBegin(GL_POINTS);
        glVertex2f(x, y);
        glEnd();
}

void display12() {

        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double t;
        glColor3f(0, 0, 0);
        glBegin(GL_POINTS);
        for (t = 0; t < 1; t = t + 0.005) {
                double xt = pow(1 - t, 3) * x1[0] + 3 * t * pow(1 - t,
2) * x1[1] + 3 * pow(t, 2) * (1 - t) * x1[2] + pow(t, 3) * x1[3];
                double yt = pow(1 - t, 3) * yc[0] + 3 * t * pow(1 - t,
2) * yc[1] + 3 * pow(t, 2) * (1 - t) * yc[2] + pow(t, 3) * yc[3];
                glVertex2f(xt, yt);

        }
        glColor3f(1, 1, 0);
        for (i = 0; i < 4; i++) {
                glVertex2f(x1[i], yc[i]);
                glEnd();
                glFlush();
        }


}
void mymouse(int btn, int state, int x, int y)
{
        if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN && flag < 4)
        {
                x1[flag] = x;
                yc[flag] = 500 - y;
                cout << " X: " << x << " Y" << 500 - y;
                glPointSize(3);
                glColor3f(1, 1, 0);
```

Pt. No. 12

Date 27/11/20

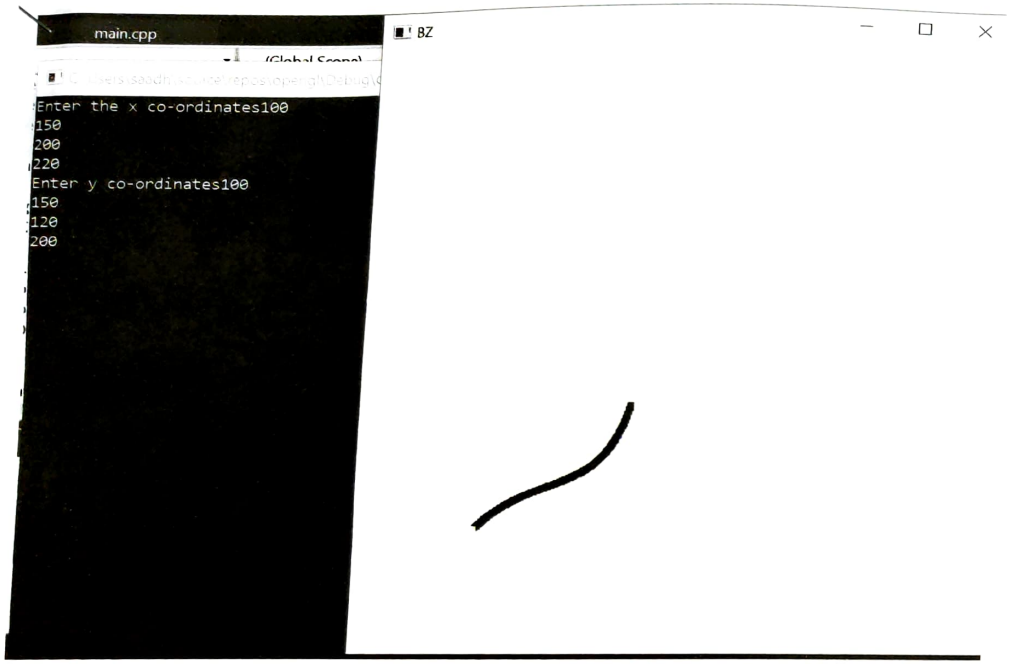
Page No.

Write a program to construct Bezier curve. Control points are supplied through keyboard/mouse.

```
#include <iostream>
#include <math.h>
#include <gl/glut.h>
using namespace std;
float f, g, r, x1[4], yc[4];
int flag = 0;
void myInit12() {
  glClearColor(1, 1, 1, 1);
  glColor3f(1, 1, 1);
  glPointSize(5);
  gluOrtho2D(0, 500, 0, 500);
}
void drawPixel(float x, float y) {
  glBegin(GL_POINTS);
  glVertex2f(x, y);
  glEnd();
}
void display12() {
  glClear(GL_COLOR_BUFFER_BIT);
  int i; double t;
  glColor3f(0, 0, 0);
  glBegin(GL_POINTS);
  for (t=0; t<=1 ; t = t + 0.005) {
  double xt = pow(1-t,3)* x1[0] + 3 * t * pow(1-
      * x1[1] + 3 * pow(t, 2) * (1-t) * x1[2] +
      pow(t, 3) * x1[3];
```

Teacher's Signature

12/18/2020 op12.png



Expt. No. Date Page No.

```
double yt = pow(1-t,3) * yc[0]+3 * t* pow (1-t, 2)*
        yc[1] +3 * pow (t,2) * (1-t) * yc[2]+
        pow (t,3) * yc[3];
  glVertex2f(xt, yt);
}
glColor3f(1, 1, 0);
for (i=0; i<4; i++){
    glVertex2f(x1[i], yc[i]);
    glEnd();
    glFlush();
} }
void mymouse (int btn, int state, int x, int y) {
  if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN
      && flag <4) {
        x1[flag] = x; yc[flag] = 500-y;
        cout << "X:" << x << "Y" << 500-y;
        glPointSize (3);
        glColor3f (1, 1, 0);
        glBegin (GL_POINTS);
        glVertex2i (x, 500-y);
        glEnd(); glFlush(); flag++;
    }
    if (flag >= 4 && btn == GLUT_ LEFT_BUTTON)
    { glColor3f (0, 0, 1); display12(); flag=0;}
}
void p12_main (int argc, char** argv)
{   glutInit (&argc, argv);
    cout << glutInitDisplayMode (GLUT_SINGLE | GLUT_RG
    glutInitWindowSize (500, 500);
```

Teacher's Signature

Date _____

Expt. No. _____ Page No. _____

```
glutInitWindowPosition(0,0);
glutCreateWindow("BZ");
glutDisplayFunc(display12);
glutMouseFunc(mymouse);
myInit12();
glutMainLoop();
}
```