NAME OF EXPERIMENT Brezenhamslin DATE

EXPERIMENT NO. 01 EXPERIMENT RESULT

PAGE NO. 1

1) Write a program to generate a line using Brezenhamis line drawing technique. Consider slopes greater than one & slopes less than one.

```
#define BLACK 0
#include <stdio.h>
#include <GL/glut.h>

int x1, x2, y1, y2;

void draw_pixel(int x, int y, int v)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void line(int x1, int x2, int y1, int y2)
{
    int dx, dy, e, i;
    int inc1, inc2, incx, incy;
    int x, y;

    dx = x2 - x1;
    dy = y2 - y1;

    if (dx < 0) dx = -dx;
    if (dy < 0) dy = -dy;
```

```
#define BLACK 0
#include <stdio.h>
#include <GL/glut.h>
int x1, x2, y1, y2;

void draw_pixel(int x,int y,int value)
{
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
}


void bres(int x1, int x2, int y1, int y2)
{
int dx,dy,i,e;
int incx,incy,inc1,inc2;
int 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)
{
draw_pixel(x,y,BLACK);
e=2*dy-dx;
inc1=2*(dy-dx);
inc2=2*dy;
for(i=0;i<dx;i++)
{
```

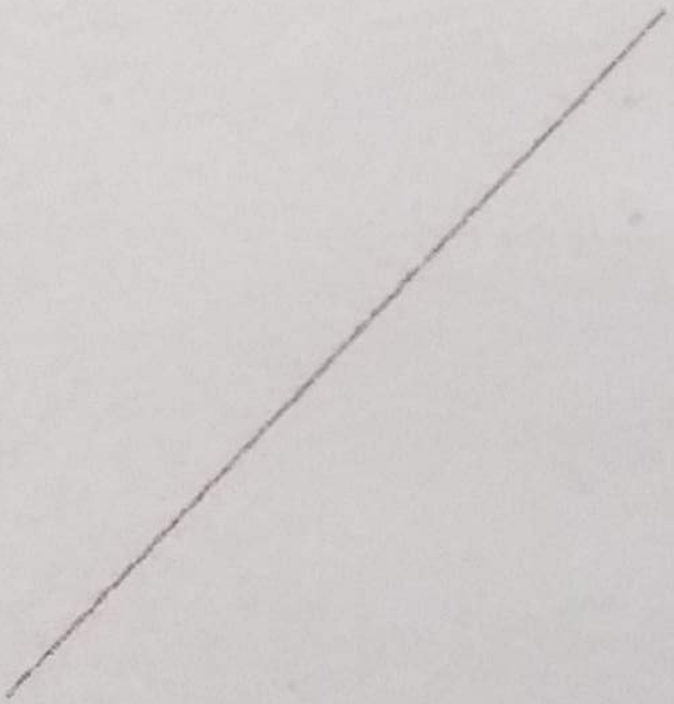
NAME OF EXPERIMENT _____ DATE _____ PAGE NO. 2

EXPERIMENT NO. _____ EXPERIMENT RESULT _____

```
incx = 1;
if (x2 < x1)   incx = -1;
incy = 1;
if (y2 < y1)   incy = -1;
x = x1;
y = y1;
if (dx > dy)
{
   draw_pixel (x, y, BLACK);
   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, BLACK);
   }
}
else
{
   draw_pixel (x, y, BLACK);
```

```
nter points: x1, y1, x2, y2
00 200 300 400
```

Bresenham,s Algorithm



NAME OF EXPERIMENT Circle & ellipse DATE

EXPERIMENT NO. 02 EXPERIMENT RESULT

2) Write a program to generate a circle and ellipse using Bresenham's circle & ellipse drawing technique

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

int xc, yc, r;
int rx, ry, xce, yce;
void drawCircle(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();
}
```

```
#include<GL/glut.h>
#include<stdio.h>
int xc, yc, r;
int 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 circlebres()
{
    glClear(GL_COLOR_BUFFER_BIT);
    int x = 0, y = r;
    int d = 3 - 2 * r;
```

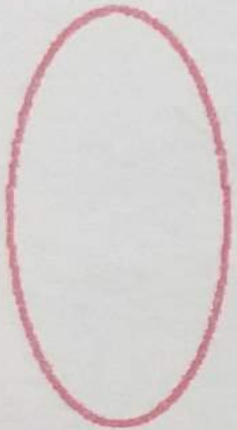
NAME OF EXPERIMENT ..... DATE ..... PAGE NO. 6

EXPERIMENT NO. ..... EXPERIMENT RESULT .....

```
void circlebres()
{
    glClear (GL_COLOR_BUFFER_BIT);
    int x=0, y=r;
    int d = 3 - 2*r;

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

void drawellipse(int xce, int yce, int rx,
                                    int ry)
{
    glBegin(GL_POINTS);
    glVertex2i (x+xce, y+yce);
    glVertex2i(-x+xce, y+yce);
    glVertex2i(x+xce, -y+yce);
    glVertex2i(-x+xce, -y+yce);
    glEnd();
}
```

nter coordinates of centre of ellipse and major and minor radius
0 50 50 100

Ellipse



nter 1 to draw circle , 2 to draw ellipse
1
nter coordinates of centre of circle and radius
100 100 50

Circle

NAME OF EXPERIMENT Sierpinski DATE

EXPERIMENT NO. 03 EXPERIMENT RESULT

3) Write a program to recursively subdivide a tetrahedron to form 3D Sierpinski gasket.

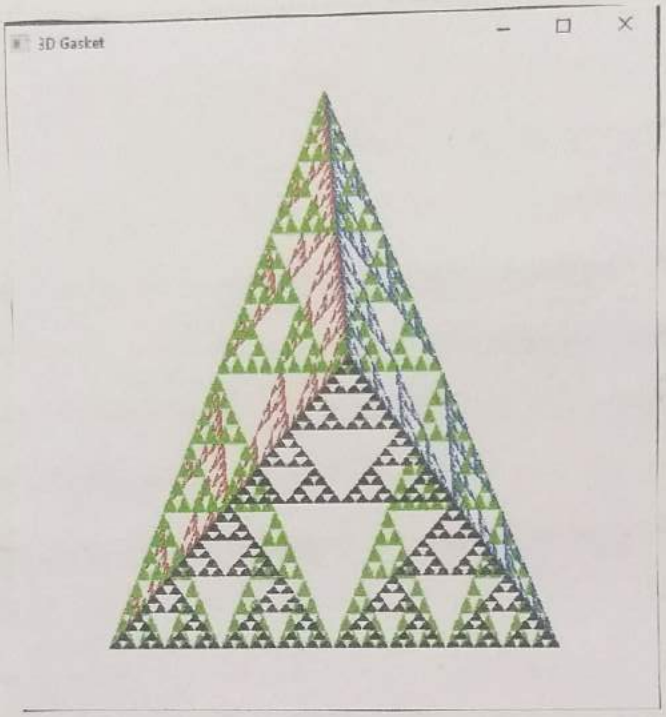
```
#include <GL/glut.h>
#include <iostream>
typedef GLfloat point[3];

point v[] = { {30.0, 50.0, 100.0},
              {0.0, 450.0, -150.0};
              {-350.0, -400.0, -150.0}.
              {350.0; -400.0; -150.0} };

int n;

void triangle (point a, point b, point
{
    glBegin(GL_triangles);
    glVertex3fv(a);
    glVertex3fv(b);
    glVertex3fv(c);
    glEnd();
}

void divide_triangle (point a, point
                      point c, int m)
{
    point v0, v1, v2;
    int j;
```

```
#include <GL/glut.h>
#include<iostream>
typedef GLfloat point[3];
point v[] = { {30.0, 50.0, 100.0}, {0.0, 450.0, -150.0},
 {-350.0, -400.0, -150.0}, {350., -400., -150.0} };
int n;
void triangle(point a, point b, point c)

{
      glBegin(GL_TRIANGLES);
      glVertex3fv(a);
      glVertex3fv(b);
      glVertex3fv(c);
      glEnd();
}
void divide_triangle(point a, point b, point c, int m)
{
      point v0, v1, v2;
      int j;
      if (m > 0)
      {
            for (j = 0; j < 3; j++) v0[j] = (a[j] + b[j]) / 2;
            for (j = 0; j < 3; j++) v1[j] = (a[j] + c[j]) / 2;
            for (j = 0; j < 3; j++) v2[j] = (b[j] + c[j]) / 2;
            divide_triangle(a, v0, v1, m - 1);
            divide_triangle(c, v1, v2, m - 1);
            divide_triangle(b, v2, v0, m - 1);
      }
      else(triangle(a, b, c));
}
void tetra(int m)
{
```

NAME OF EXPERIMENT _____ DATE _____ PAGE NO. 12

EXPERIMENT NO. _____ EXPERIMENT RESULT _____

```
if (m >0)
{
  for(j=0;j<3;j++)
        v0[j] = (a[j] + b[j]) /2;
  for(j=0;j<3;j++)
        v1[j] = (a[j] + c[j]) /2;
  for(j=0;j<3;j++)
        v2[j] = (b[j]+c[j]) /2;

  divide_triangle(a, v0, v1, m-1)
  divide_triangle(c, v1, v2, m-1)
  divide_triangle(b, v2, v0, m-1);
}

else
    triangle(a, b, c);
}

void tetra(int m)
{
  glColor3f(1.0, 0.0, 0.0);
  divide_triangle(v[0], v[1], v[2], m);
  glColor3f(0.0, 1.0, 0.0);
  divide_triangle(v[3], v[2], v[1], m);
  glColor3f(0.0, 0.0, 1.0);
  divide_triangle(v[0], v[3], v[1], m);
  glColor3f(0.0, 0.0, 0.0);
  divide_triangle(v[0], v[2], v[3], m);
}
```


3D Gasket

NAME OF EXPERIMENT Scanline fill DATE

EXPERIMENT NO. 04 EXPERIMENT RESULT

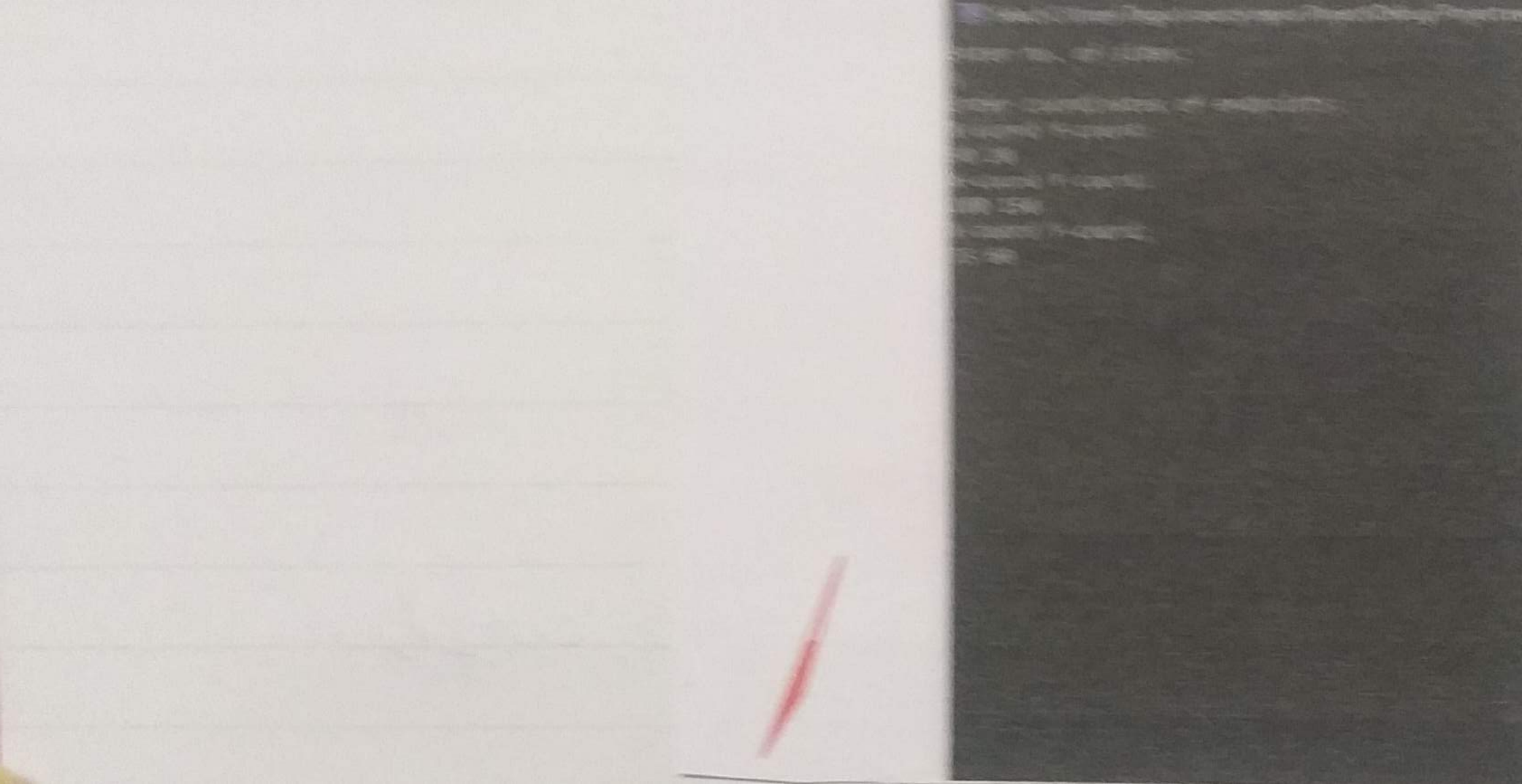
4) Write a program to fill any given polygon using scanline area filling algorithm.

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

using namespace std;
float x[100], y[100];

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

void drawline(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();
}
```

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

using namespace std;
float x[100], y[100];

int n, m;
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);
```



```
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[m++] = x1 + (scanline - y1)*
                (x2-x1)/(y2-y1);
}

void scanfill (float x[], float y[])
{
  for(int s1=0; s1<=wy; s1++)
  {
    m = 0;
    for(int i=0; i<n; i++)
    {
      edgeDetect (x[i], y[i], x[(i+1)%n],
                  y[(i+1)%n], s1);
    }
    sort (intx, intx+m);
    if (m >= 2)
      for(int i=0; i<m; i+=2)
        drawline (intx[i], s1,
                  intx[i+1], s1);
  }
}
```



NAME OF EXPERIMENT House DATE

EXPERIMENT NO. 05 EXPERIMENT RESULT

5) Write a program to create a house like figure and
i) Rotate it about a given fixed point
ii) Reflect it about an axis $y = mx + c$

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

float house[11][2] = { {100,200}, {200,250},
    {300,200}, {100,200}, {100,100},
    {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]);
```

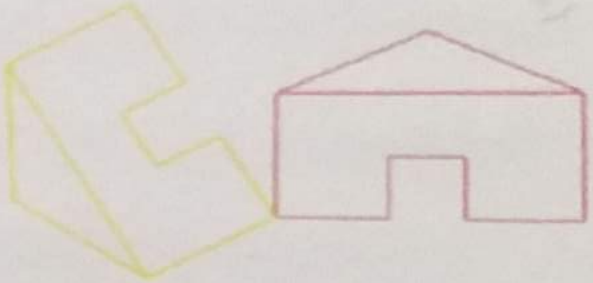
```
#include<gl/glut.h>
#include <math.h>
#include<stdio.h>


float house[11][2] = { { 100,200 },{ 200,250 },{ 300,200 },{ 100,200 },{
100,100 },{ 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);
      glRotatef(angle, 0, 0, 1);
      glTranslatef(-100, -100, 0);
      glColor3f(1, 1, 0);
      glBegin(GL_LINE_LOOP);
      for (int i = 0; i < 11; i++)
```

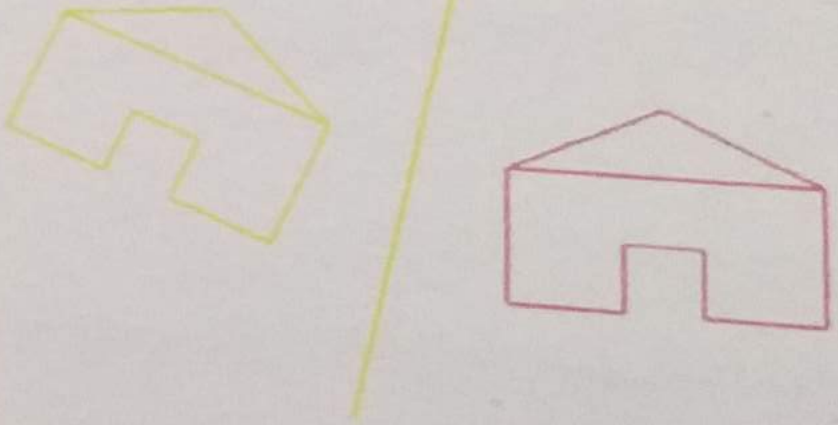
```
glutMouseFunc(mouse);
myInit();
glutMainLoop();
}
```

NAME OF EXPERIMENT _____ DATE _____ PAGE NO. 20

EXPERIMENT NO. _____ EXPERIMENT RESULT _____

```
      glEnd();
      glFlush();

      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);
      glLoadIdentity();
      glColor3f(1,0,0);
      glBegin(GL_LINE_LOOP);
      for(int i=0; i<11; i++)
            glVertex2fv(house[i]);
      glEnd();
      glFlush();
```





NAME OF EXPERIMENT Line Clip DATE PAGE NO. 24

EXPERIMENT NO. 06 EXPERIMENT RESULT

6) Write a program to implement the Cohen - Sutherland line clipping algorithm.

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

#define outcode 1
#define true 1
#define false 0

double xmin, ymin, xmax, ymax;
double xvmax, xvmin, yvmin, yvmax;

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

int n;

struct lineseqment
{
    int x1;
    int x2;
    int y1;
    int y2;
};
struct lineseqment ls[10];
```

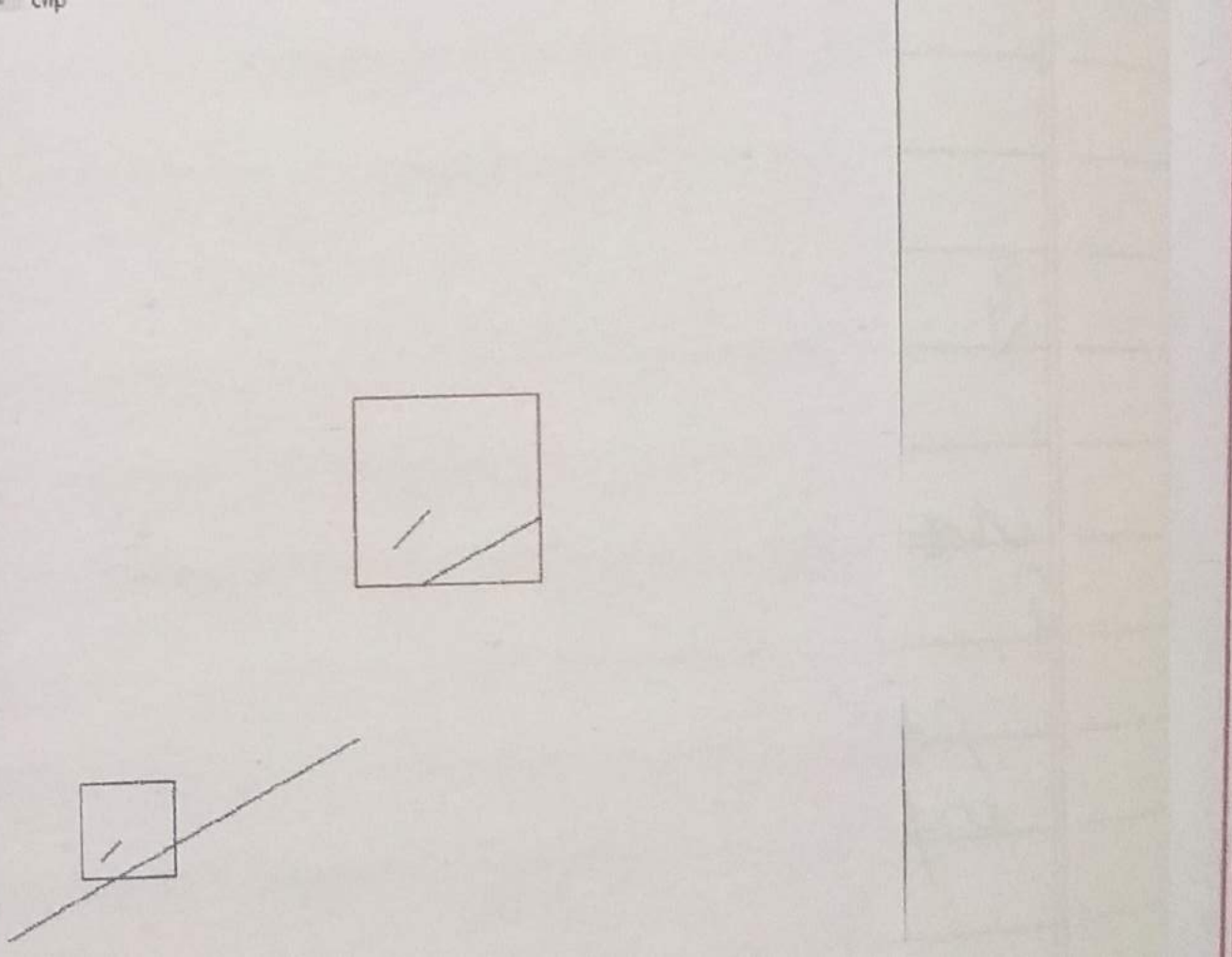
```
#include<stdio.h>
#include<stdlib.h>
#include<gl/glut.h>

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


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

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

outcode computeoutcode(double x, double y)
{
```

NAME OF EXPERIMENT _____ DATE _____ PAGE NO. 215

EXPERIMENT NO. _____ EXPERIMENT RESULT _____

```
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 oc0, oc1, oc;
  bool accept = false, done = false;

  oc0 = computeoutcode (x0, y0);
  oc1 = computeoutcode (x1, y1);

  do
  {
    if (!(oc0 | oc1))
    {
      accept = true;
      done = true;
    }
```

```
nter window coordinates (xmin ymin xmax ymax):
0 50 100 100
nter viewport coordinates (xvmin yvmin xvmax yvmax) :
00 200 300 300
nter no. of lines:
2
nter line endpoints (x1 y1 x2 y2):
0 60 70 70
nter line endpoints (x1 y1 x2 y2):
0 20 200 120
```



NAME OF EXPERIMENT Liang Barsky DATE

EXPERIMENT NO. 07 EXPERIMENT RESULT

7) Write a program to implement Liang - Barsky line clipping algorithm

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

double xmin, ymin, xmax, ymax, xvmin,
       xvmax, yvmin, yvmax
int n;

struct line_segment
{ int x1;
  int x2;
  int y1;
  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);
  }
```

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

double xmin, ymin, xmax, ymax; //50 50 100 100
double xvmin, yvmin, xvmax, yvmax; //200 200 300 300

int n;

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;  // 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 t1
            {
                  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
```

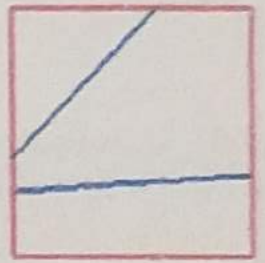
NAME OF EXPERIMENT _____ DATE _____

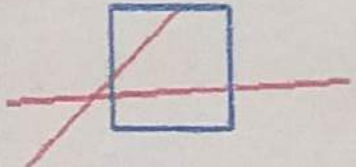
EXPERIMENT NO. _____ EXPERIMENT RESULT _____ PAGE NO. 32

```
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 LiangBarsky (double x0, double y
                  double x1, double
{
    double dx = x1 - x0, dy = y1 - y0,
           u1 = 0.0, u2 = 1.0;
    glColor3f (1.0, 0.0, 0.0);
    glBegin (GL_LINE_LOOP);
    glVertex2f (xmin, ymin);
    glVertex2f (xmax, ymin);
    glVertex2f (xmax, ymax);
    glVertex2f (xmin, ymax);
    glEnd();
    if (clipTest (-dx, x0 - xmin, &u1,
        if (clipTest (dx, xmax - x0, &u1, &
            if (clipTest (-dy, y0 - ymin, &u1, &
```

```
nter window coordinates: (xmin ymin xmax ymax)
0 50 100 100
nter viewport coordinates: (xvmin yvmin xvmax yvmax)
00 200 300 300
nter no. of lines:

nter coordinates: (x1 y1 x2 y2)
 60 150 70
nter coordinates: (x1 y1 x2 y2)
0 30 80 100
```

Liang Barsky Line Clipping Algorithm





NAME OF EXPERIMENT Polyclip DATE

EXPERIMENT NO. 08 EXPERIMENT RESULT

8) Write a program to implement the Cohen - Hodgeman polygon clipping algorithm

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

using namespace std;
int poly_size, poly_points[20][20], output,
    output_points[20][2], clip_size, clip_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;
}
```

```
#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);
```

NAME OF EXPERIMENT ..... DATE .....

EXPERIMENT NO. ..... EXPERIMENT RESULT ..... PAGE NO. 38

```
int y_intersect(int x1, int y1, int x2,
      int y2, int x3, int y3, int
      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 -

return num/den;
}

void clip(int ppoints[][2], int &ppsize,
      int x1, int y1, int x2, int y2)
{
int npoints[MAXPOINTS][2], npsize = 0, i;
for(i=0; i< ppsize; i++)
{
  int k = (i+1) % ppsize;
  int ix = ppoints[i][0], iy = ppoints[i][
  int kx = ppoints[k][0], ky = ppoints[k][

int i_pos = (x2-x1) * (iy - y1) - (y2-y1) * (ix-x
int k_pos = (x2-x1) * (ky - y1) - (y2-y1) * (kx - x

if(i_pos >= 0 && k_pos >= 0)
{
  npoints[npsize][0] = kx;
  npoints[npsize][1] = ky;
  npsize ++;
}
```

Polygon Clipping

```
Enter no. of vertices:
3
Polygon Vertex:
50 70
Polygon Vertex:
100 50
Polygon Vertex:
90 120
Enter no. of vertices of clipping window:4
Clip Vertex:
80 80
Clip Vertex:
120 90
Clip Vertex
115 110
Clip Vertex:
80 110
```

NAME OF EXPERIMENT Car DATE

EXPERIMENT NO. 09 EXPERIMENT RESULT PAGE NO. 42

9) Write a program to model a car like figure using display lists.

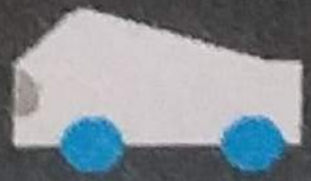
```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>

#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, 75, 0);
    glEnd();
    glEndList();
}

void wheelist()
{
    glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
```

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

NAME OF EXPERIMENT _____ DATE _____

EXPERIMENT NO. _____ EXPERIMENT RESULT _____ PAGE NO. 43

```
      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 myinit()
{
  glClearColor (0, 0, 0, 0);
  gluOrtho (0, 600, 0, 600, 0, 600);
}

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



NAME OF EXPERIMENT Cube spin DATE

EXPERIMENT NO. 10 EXPERIMENT RESULT

10) Write a program to create a color cube and spin it

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

GLfloat vertices[] = { -1, -1, -1, -1, -1, 1, -1, 1,
                       -1, -1, 1, -1, 1, 1, 1, 1,
                       -1, 1, 1, 1, 1, -1, 1, -1 };

GLfloat normals[] = { -1, -1, -1, -1, -1, 1, -1, 1,
                      -1, -1, 1, -1, 1, 1, 1, 1,
                      -1, 1, 1, 1, 1, -1, 1, -1 };

GLbyte 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 };

GLfloat colors[] = { 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0,
                     0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1 };

static GLfloat theta[] = { 0, 0, 0 };
static GLfloat beta[] = { 0, 0, 0 };
static GLint axis = 2;
```

```
#include <stdlib.h>
#include <GL/glut.h>
#include<gl\GL.h>
#include<gl\GLU.h>
#include <time.h>
#include"Header.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)
{
}
```

```
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST); /* Enable hidden--surface--removal */
    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();
}
```

NAME OF EXPERIMENT _____ DATE _____

EXPERIMENT NO. _____ EXPERIMENT RESULT _____ PAGE NO. 47

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

void display (void)
{
    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();
}

void spinCube()
{
    delay(0.01);
    theta[axis] += 2.0;
    if (theta[axis] > 360.0) theta[axis] -= 360;
    glutPostRedisplay();
}
```

NAME OF EXPERIMENT Curves DATE

EXPERIMENT NO. 11 EXPERIMENT RESULT

11) Create a menu with 3 entries curves, color, quit: Curve has a submenu with four entries Limacon, Cardiod, Three-leaf & spiral: Color submenu has all 8 RGB colors.

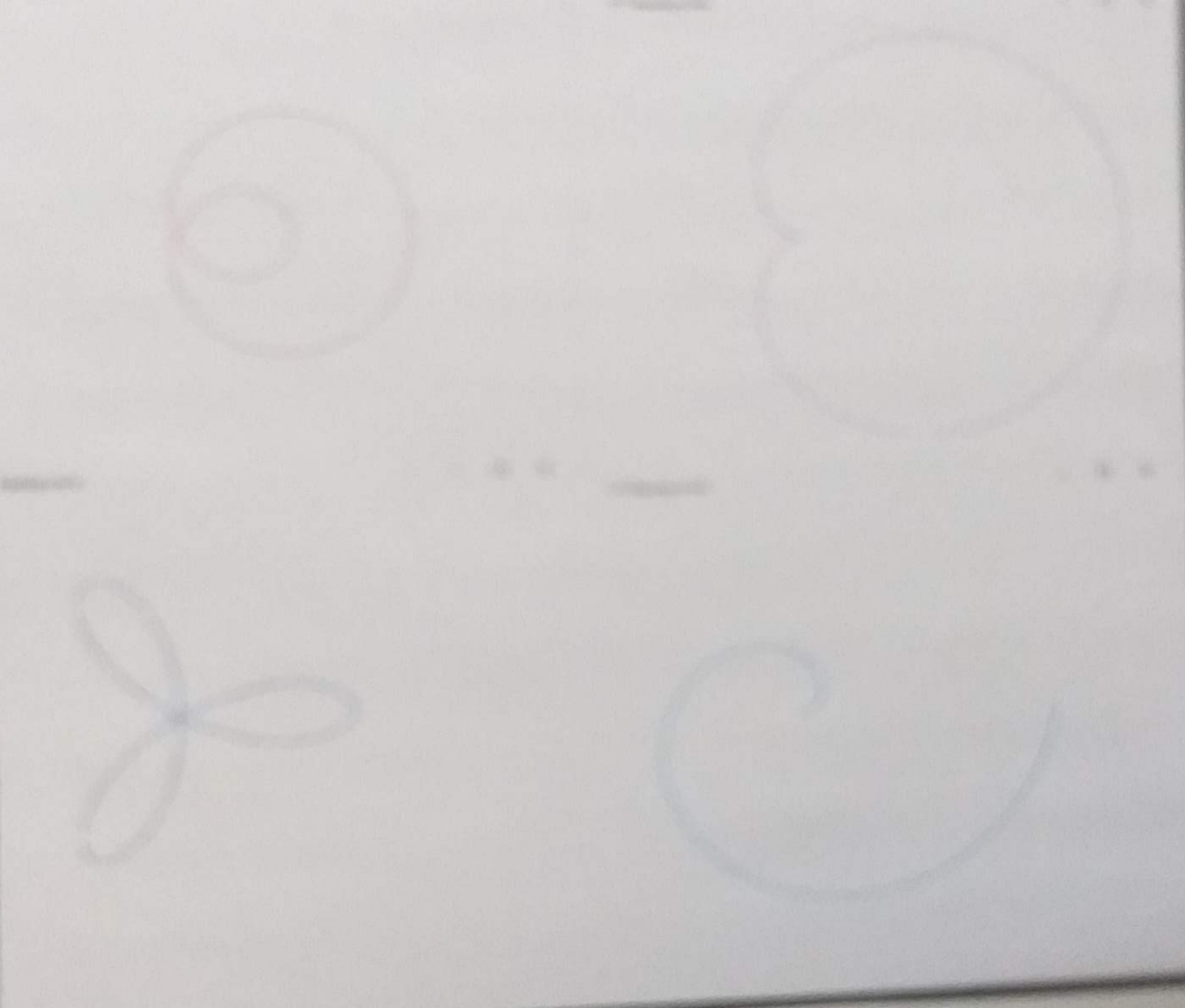
```
#include <gl/glut.h>
#include <math.h>
#include <stdio.h>

struct Screenpt
{
    int x;
    int y;
};
typedef enum{ limacon = 1, cardiod = 2,
              threeleaf = 3, spiral = 4 } curveName;

int w = 600, h = 700; red = 0, green = 0, blue = 0;
int curve = 1;

void myinit()
{
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
```

```
#include<gl/glut.h>
#include<math.h>
#include<stdio.h>
#include"Header.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 myinit(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;
```

NAME OF EXPERIMENT ..... DATE ..... PAGE NO. 51

EXPERIMENT NO. ..... EXPERIMENT RESULT .....

```
void lineSegment (ScreenPt P1, ScreenPt P2)
{
  glBegin (GL_Line),
  glVertex2i (P1.x, P1.y);
  glVertex2i (P2.x, P2.y);
  glEnd();
  glFlush();
}

void drawcurve (int curvenum)
{
  const double 2Pi = 6.283185;
  const int a=175, b=60;
  float r, theta, dtheta = 1.0/float(a);
  int x0=200, y0=260;
  ScreenPt curvept[2];
  curve = curvenum;
  glColor3f (red, green, blue);
  curvept[0].x = x0;
  curvept[1].y = y0;
  glClear (GL_COLOR_BUFFER_BIT);
  switch (curvenum)
  {
    case limacon : curvept[0].x += a+b;
                   break;
    case cardiod : curvept[0].x += a+a;
                   break;
    case threeleaf : curvept[0].x += a;
                   break;
    case spiral : break;
    default : break;
  }
```



NAME OF EXPERIMENT Bezier Curve DATE

EXPERIMENT NO. 12 EXPERIMENT RESULT PAGE NO. 56

12) Write a program to construct Bezier curve

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

using namespace std;
float x, y, z, x1[4], y1[4];
int flag = 0;

void myinit()
{
    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();
}
```

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

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

      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 display() {

      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) *
```

NAME OF EXPERIMENT _____ DATE _____

EXPERIMENT NO. _____ EXPERIMENT RESULT _____ PAGE NO. 57

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    int i;
    double t, xt, yt;
    glColor3f(0,0,0);
    glBegin(GL_POINTS);
    for(t=0; t<1; t=t+0.002)
    {
      xt = pow(1-t,3)*x1[0] + 3*t*pow(1-t,
        +3*pow(t,2)*(1-t)*x1[2] +
          pow(t,3)*x1[3];

      yt = pow(1-t,3)*y1[0]+3*t*pow(1-t,2)*
        + 3*pow(t,2)*(1-t)*y1[2] +
          pow(t,3)*y1[3];

      glVertex2f(xt, yt);
    }
    glColor3f(1,1,0);
    for(i=0; i<4; i++)
    {
      glVertex2f(x1[i], y1[i]);
      glEnd();
      glFlush();
    }

}
```