Date _____

Pgt. No. 1 Page No. 1

Write a program to generate a line using Bresenham's line drawing technique. Consider slopes greater than one and slopes less than one. User can specify inputs through keyboard / mouse.

```
#define BLACK 0
#include <stdio>
#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 lbres(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;
 inc = 1;
 if (x2<x1) incx = -1;
 incy = 1;
 if (y2<y1) incy = -1;
 x = x1; y = y2;
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. 2

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

Teacher's Signature _____

```
else x+= inc2;
  y+= incy;
  draw - pixel(x, y, 0);
  }
 }
}

void display() {
 glClear(GL_COLOR_BUFFER_BIT);
 bres(x1, y1, x2, y2);
 glFlush();
}

void myinit() {
 glClearColor(1.0, 1.0, 1.0, 1.0);
 glColor3f(1.0, 0.0, 0.0);
 glPointSize(1.0);
 glMatrixMode(GL_PROJECTION);
 gloadIdentity();
 gluOrtho2D(0.0, 499.0, 0.0, 499.0); }

void main(int argc, char** argv)
{
  printf("Enter points: x1 y1 x2 y2");
  scanf("%d %d %d %d", &x1, &x2, &y1, &y2);
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("Bresenham's Algorithm");
```

Teacher's Signature _____

```
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

Date

Expt. No. 4

Page No. 5

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 xc, yc, r, rx, ry, xcc, ycc;

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;
```

Teacher's Signature

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

draw_circle(xc, yc, x, y); }
glFlush();
}

int p1_x, p2_x, p1_y, p2_y;
int point1_done = 0;
void myMouseFunc(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)
{  p2_x = x - 250;
   p2_y = 250 - y;
   xc = p1_x;
   yc = p1_y;
   float rsq = (p2_x - p1_x)*(p2_x - p1_x) + (p2_y - p1_y)*(p2_y - p1_y);
   r = (int)(sqrt(rsq));
   circlebres();
   point1_done = 0;
} }
```

```
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 midpointellipse()
{
    glClear(GL_COLOR_BUFFER_BIT);
    float dx, dy, d2, d1, x, y;
    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)
{
    if (d1<0) {
        x++;
        dx = dx + (2*ry*ry);
        d1 = d1+ dx + (ry*ry);
    } else {
        x++;
        y--;
        dx = dx + (2*ry*ry);
        dy = dy - (2*rx*rx),
        d1 = d1 + dx - dy + (ry*ry),
    }
}
```

Teacher's Signature

Date _____

Expt. No. _____ Page No. _____

```
d2 = ((dy * dy)) * (x+0.5)*(x+0.5))) + ((dx*dx)*(y-1)*(y-1))) -
     (dx*dx*dy*dy);

while (y>=0){
draw_ellipse(rx, ry, x, y);
if (d2>0)
{ y--;
  dy = dy - (2*dx*dx),
  d2 = d2 + (dx*dx) - dy;
}else
{  y--;
   x++,
   dx = dx + (2*dy*dy);
   dy = dy - (2*dx*dx),
   d2 = d2 + dx - dy + (dx*dx);
   }
}
   glFlush();
}
int p1c_x, p2c_x, p1c_y, p2c_y, p3c_x, p3c_y;
int point1c_done = 0;
void myMouseFunc(int button, int state, int x, int y){
 if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && point1c_done==0)
{   p1c_x = x - 250)
    p1c_y = 250 - y;
    rx = p1c_x;
    ry = p1c_y;
    point1c_done = 1;
```

Teacher's Signature _____

```
}
else if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && pointlc-
done == 1) {
p2c_x = x-250;
p2c_y = 250-y;
float inp = (p2c_x - p1c_x) * (p2c_x - p1c_x) + (p2c_y - p1c_y) * (p2c_y-p1c_y);
rx = (int) (sqrt(inp));
pointlc_done = 2;
}
else if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && pointlc.done
== 2) {
p3c_x = x-250;
p3c_y = 250-y;
float inp = (p3c_x - p1c_x) * (p3c_x - p1c_x) + (p3c_y - p1c_y) * (p3c_y -
p1c_y);
ry = (int)(sqrt(inp));
midbillips();
pointlc_done = 0;
} }
void minit() {
glClearColor(1,1,1,1);
glColor3f (1.0, 0.0, 0.0);
glPointSize (3.0);
gluOrtho2D (-250, 250, -250, 250);
}
void main(int argc, char* argv[]) {
glutInit (&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
```

Teacher's Signature _____

```
glutInitWindowPosition (0,0);
int id1 = glutCreateWindow ("Circle");
glutSetWindow (id1);
glutMouseFunc (myMouseFunccircle);
glutDisplayFunc (myDrawing);
menu ();
glutInitWindowSize (500, 500);
glutInitWindowPosition (600, 100);
int id2 = glutCreateWindow ("Ellipse");
glutSetWindow (id2);
glutMouseFunc (myMouseFunc);
glutDisplayFunc (myDrawing);
printf ("Enter 1 to draw circle, 2 ellipse \n");
int ch;
scanf ("%d", &ch);
switch (ch) {
case 1: printf ("Enter coordinates of circle & radius \n");
        scanf ("%d %d %d", &xc, &yc, &r);
        glutCreateWindow ("Circle");
        glutDisplayFunc (circledraw);
        break;
case 2: printf ("Enter coordinates of centre of ellipse and major and minor
                radius \n");
        scanf ("%d %d %d %d", &xc, &yc, &rx, &ry);
        glutCreateWindow ("Ellipse");
        glutDisplayFunc (midellipse); break;
}
menu();
glutMainLoop(); }
```

Teacher's Signature _____

Expt. No. 3 Page No.

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

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<GL/glut.h>
using namespace std;
int n;
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;
```

Teacher's Signature

Date _____

Expt. No. _____ Page No. _____

```
    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);
  glFlush();
}
void myinit() {
  glClearColor(1,1,1.0,1.0);
  glColor3f(1.0,0,0);
  glPointSize(5.0);
  glOrtho(-500, 500, -500, 500, -500, 500);
}
```

Teacher's Signature _____

Date _____

t. No. _____

Page No. _____

```
int main(int argcp, char** argv){
cout<<"Enter m:";
cin>>m;
glutInit(&argcp, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_DEPTH);
glutInitWindowSize(300,300);
glutInitWindowPosition(100,100);
glutCreateWindow("Dis");
glutDisplayFunc(tilrahudon);
glEnable(GL_DEPTH_TEST);
myinit();
glutMainLoop();
}
```

Teacher's Signature _____

Date

Pgt. No. 4

Page No.

Write a program to fill a polygon using scan line 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 int x[10] = {0};

void drawline(float x1, float y1, float x2, float y2){
glFlush(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)
int x[m++] = x1 + (scanline-y1)*(x2-x1)/(y2-y1);
}
```

Teacher's Signature

```
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(intn, (intn+m));
 if (m>=2)
  for(i=0; i<m;) i+=2)
   draw_line (intn[i], s1, intn[i+1], s1); } }

void display_filled_polygon() {
 glClear(GL_COLOR_BUFFER_BIT);
 glLineWidth(2);
 glBegin(GL_LINE_LOOP);
for(int i=0; i<n; i++)
   glVertex2f (x[i], y[i]);
 glEnd();
 scanfill(x,y);
}
void myInit() {
glClearColor (1,1,1,1);
glColor3f (0,0,1);
glPointSize (1);
gluOrtho2D (0,wx, 0,wy);
}
void main(int ac, char* av[]) {
glutInit(&ac, av);
printf("No of sides");
scanf_s ("%d", &n);
```

Date _____

Expt. No. _____ Page No. _____

```
for(int i=0;i<n;i++){
printf("x-coord y-coord");
scanf("%f %f",&x[i],&y[i]);}
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("scanline");
glutDisplayFunc(display_filled_polygon);
myInit();
glutMainLoop();
}
```

Teacher's Signature _____

Date _____

Expt. No. 5 Page No. _____

Write a program to create a house like figure and perform the following operation

i) Rotate it about a given fixed point

ii) Reflect it about an axis $y = mx + c$

```
#include <GL/glut.h>
#include <GL/freeglut.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();
glOrtho2D(-450,450,-450,450);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glColor3f(1,0,0);
glBegin(GL_LINE_LOOP);
for(int i=0;i<7;i++)
  glVertex2fv(house[i]);
glEnd();
glFlush();
glPushMatrix();
glTranslatef(100,100,0);
glRotatef(angle,0,0,1);
glTranslatef(-100,-100,0);
```

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. _____

```
glColor3f(1,0,1);
glBegin(GL_LINE_LOOP);
for(int i=0; i<7; 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();
glOrtho2D(-450,450,-450,450);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glColor3f(1,0,0);
glBegin(GL_LINE_LOOP);
for(int i=0;i<7;i++)
  glVertex2fv(house[i]);
glEnd();
glFlush();
float x1=0, x2=500, y1=m*x1+c, y2=m*x2+c;
glColor3f(1,0,1);
glBegin(GL_LINES);
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glEnd();
glFlush();
```

Teacher's Signature _____

Expt. No. _____ Page No. _____

```
glPushMatrix();
glTranslatef(0,c,0);
thela = atan(m);
theta = thela*180/3.14;
glRotatef(thela,0,0,1);
glScalef(1,-1,1);
glRotatef(-thela,0,0,1);
glTranslatef(0,-c,0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<7; i++) glVertex2fv(house[i]);
glEnd(); glPopMatrix(); glFlush(); }
void myInit(){
glClearColor(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 stat, int x, int y){
if(btn==GLUT_LEFT_BUTTON && stat==GLUT_DOWN){ display(); }
else if(btn==GLUT_RIGHT_BUTTON && stat==GLUT_DOWN){ display(); } }

void main(int argc, char** argv){
printf("Angle");
scanf("%d",&angle);
printf("Enter c and m value");
scanf("%f %f",&c,&m);
```

Teacher's Signature _____

Date _____

Page No. _____

```
glInit(&argc, argv);
glInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glInitWindowSize(900, 900);
glInitWindowPosition(100, 100);
glCreateWindow("House");
glDisplayFunc(display);
glMouseFunc(mouse);
Init();
glMainLoop();
```

Teacher's Signature _____

Date _____

Expt. No. 5 Page No. _____

Write a program to create a house like figure using scan line algorithm
i) Rotate it about a given fixed point using OpenGL transformation
ii) Reflect it about an axis $y=mx+c$ using OpenGL transformation functions

```
#include <GL/glut.h>
#include <GL/gluglut.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;
void display()
{
  glClearColor(1,1,1,0);
  glClear(GL_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
  glOrtho2D(-450, 450, -450, 450);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glColor3f(1,0,0);
  glBegin(GL_LINE_LOOP);
  for(int i=0; i<7; i++)
    glVertex2fv(house[i]);
  glEnd();
  glPopMatrix();
  glFlush();
}
void display2(){
 glClearColor(1,1,1,0);
```

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

```
glClear(GL_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho2D(-450, 450, -450, 450);
glMatrixMode(GL_MODELVIEW);
glColor3f(1,0,0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<7; i++)
    glVertex2fv(house[i]);
glEnd();
glFlush();
float x1=0, x2=500, y1=m*x1+c, y2=m*x2+c;
glColor3f(1,0,1);
glBegin(GL_LINES);
glVertex2f(x1,y1);
glVertex2f(x2,y2);
glEnd();
glFlush();
glPushMatrix();
glTranslatef(0,c,0);
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<7; i++)
    glVertex2fv(house[i]);
glEnd();
```

Teacher's Signature _____

```
    glPopMatrix();
    glFlush();
}
void myInit(){
    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){
   display(); }
 else if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN){
   display2(); }}
void main(int argc, char** argv){
  printf("Enter rotation angle\n");
  scanf("%d",&angle);
  printf("Enter c and m value");
  scanf("%f %f",&c,&m);
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(900,900);
  glutCreateWindow("House Rotation");
  glutDisplayFunc(display);
  glutMouseFunc(mouse);
  myInit();
  glutMainLoop(); }
```

Date

Expt. No. 6

Page No.

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

```
#include<stdio.h>
#include<stdlib.h>
#include<GL\glut.h>
#include<GL\freeglut.h>
#include<iostream>
#define outcode int
#define true 1
#define false 0
GLfloat xmin, ymin, xmax, ymax, xvmin, yvmin, xvmax, yvmax;
const int RIGHT=4, LEFT=8, TOP=1, BOTTOM=2;
int n;
struct line_segment{
int x1, y1, x2, 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

```
void cohensutherl(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(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);
 }
```

Date _____

Expt. No. _____ Page No. _____

```
else {
    x1 = x;
    y1 = y;
    outcode1 = computeOutCode(x1, y1);
    }
  }
} while(!done);
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(0.1, 1, 0);
  glVertex2d(vx0, vy0);
  glEnd();
 }
}

void display(){
  glClear(GL_BUFFER_BIT);
  glColor3f(0.0, 0, 1);
  glBegin(GL_LINE_LOOP);
  glVertex2f(xmin, ymin);
  glVertex2f(xmax, ymin);
  glVertex2f(xmax, ymax);
  glVertex2f(xmin, ymax);
  glEnd();
```

Teacher's Signature _____

```
glColor3f(0,1,1);
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();
}
void display()
{
glClear(GL_BUFFER_BIT);
glColor3f(0,0,1);
glBegin(GL_LINE_LOOP);
glVertex2f(xmin, ymin);
glVertex2f(xmax, ymin);
glVertex2f(xmin, ymax);
glEnd();
}
void myinit()
{
glClearColor(1,1,1,1);
glColor3f(1,0,0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,500,0,500);
}

void main(int argc, char** argv)
{
```

```
printf ("Enter window coordinates");
cin >> xmin >> ymin >> xmax >> ymax;
printf ("Enter viewport coordinates ");
cin >> vxmin >> vymin >> vxmax >> vymax;
printf ("Enter no of lines ");
cin >> n;
for(int i=0; i<n; i++)
 printf ("Enter line endpoints (x1 y1 x2 y2): \n");
 scanf ("%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 (600, 600);
glutCreateWindow ("clip");
myinit ();
glutDisplayFunc (display);
glutMainLoop ();
}
```

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

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

double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;
int n;
struct line_segment {
 int x1;
 int y1;
 int x2;
 int y2; y;
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)
   {
```

Teacher's Signature _____

```
    if (q<0.0) return(false);
  }
return (true);
}
void LiangBarskyLineClip (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);
 glVertex2f(xvmin, yvmin);
 glVertex2f (xvmax, yvmin);
 glVertex 2f (xvmax, yvmax);
 glVertex 2f (xvmin, yvmax);
 glEnd();
 if (cliptest(-dx, x0, -xmin, &u1, &u2))
  if (cliptest (dx, xmax, -x0, &u1, &u2))
   if (cliptest(-dy, y0, -ymin, &u1, &u2))
    if (cliptest (dy, ymax, -y0, &u1, &u2)) {
    if (u2<1.0)
    {
      x1 = x0 + u2*dx;
      y1 = y0 + u2*dy;
    }
     if (u1>0.0)
    {
      x0 = x0 + u1*dx;
      y0 = y0 + u1*dy;
    }
```

Teacher's Signature _____

```
double sx = (xvmax - xvmin) / (xmax - xmin);
double sy = (yvmax - yvmin) / (ymax - ymin);
double vx0 = xvmin + (x0 - xmin) * sx;
double vx1 = xvmin + (x1 - xmin) * sx;
double vy1 = yvmin + (y1 - ymin) * 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<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++)
{
LiangBarskyLineClip (ls[i].x1, ls[i].y1, ls[i].x2, ls[i].y2);
}
```

```
glFlush();
}

void myinit()
{
glClearColor(1.0, 1.0, 1.0, 1.0);
glColor3f(1.0, 0.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho2D(0.0, 499.0, 0.0, 499.0);
}

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
printf("Enter window coordinates \n");
scanf("%f%f%f%f", &xmin, &ymin, &xmax, &ymax);
printf("Enter viewport coordinates \n");
printf("Enter no of lines: \n");
scanf("%d", &n);

for(int i=0; i<n; i++)
{
printf("Enter coordinates \n");
```

```
scanf ("%d %d %d %d", &ls[i].x1, &ls[i].y1, &ls[i].x2, &ls[i].y2);

glutCreateWindow ("Algo");
glutDisplayFunc(display);
myInit();
glutMainLoop();
```

Teacher's Signature _____

Write a program to model a car like figure using display list and move a car from one end of screen to other end. User able to control 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(0,55,0);
 glEnd();
 glEndList();
}

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

Teacher's Signature _____

```
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);
  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);
glCallList(WHEEL);
```

Teacher's Signature

```
glPopMatrix();
glTranslatef(75.25, 0.0);
glCallList(WHEEL);
glPopMatrix();
glFlush();
}
void myDisp() {
glClear(GL_COLOR_BUFFER_BIT);
carList();
moveCar(s);
wheelList();
}

void mouse(int btn, int state, int x, int y) {
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
    st = 5;
    myDisp();
}
else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
    st = 2;
    myDisp();
  }
}

int main(int argc, char* argv[]) {
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

Teacher's Signature _____

```
glutInitWindowPosition(100,100);
glutCreateWindow("CAL");
myInit();
glutDisplay(myDisp);
glutMouseFunc(mouse);
glutKeyboardFunc(mykey);
glutMainLoop();
}
```

Teacher's Signature _____

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

```
#include<stdlib.h>
#include<GL/glut.h>
#include<gl\GL.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,
1.0,1.0,0.0,1.0,0.0,0.0,0.0,1.0,
1.0,0.0, 1.0,1.0,1.0,1.0,0.0,1.0,1.0};

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,
0.0,1.0,1.0,1.0,1.0,0.0,1.0,0.0};

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

Teacher's Signature _____

```
void displaySingle(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] -= 360.0;
  glutPostRedisplay();
}

void mouse(int btn, int state, int x, int y)
{
  if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
```

```
if (btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
}

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

void main (int argc, char** argv)
{
  glutInit (&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition (100,100);
  glutInitWindowSize (500,500);
  glutCreateWindow ("colorcube");
  glutReshapeFunc (spinCube);
```

Teacher's Signature _____

```
glMouseFunc(mouse);
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();
}
```

Create a menu with three entries named curves, colors and quit. The entry curves has a sub-menu which has four entries namely Limacon, Cardiod, Three-Leaf and Spiral. Colour menu has sub menu with all 8 colors of RGB. Write a program to create above hierarchial menu

```
#include<gl/glut.h>
#include<math.h>
#include<stdio.h>
struct Screen{
int x;
int y;
};
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 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;
```

Teacher's Signature _____

```
Screen curve[2];
curve = curvenum;
glColor3f (red, green, blue);
curveP[0].x = x0;
glClear ( GL_COLOR_BUFFER_BIT);
Switch (curveNum) {
case limacon : curveP[0].x += a+b; break;
case cardoid : curveP[0].x += a+a; break;
case threeleaf : curveP[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 cardoid : 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

```
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 3:  red = 0,
        green = 0;
        blaue :=0;
        break;
case 4:  red=1;
        green = 0;
        blue = 1;
        break;
case 5.  red = 0;
        green = 1;
        blue = 1;
        break;
case 6:  red = 1;
        green = 1;
        blue = 0;
        break;
```

Teacher's Signature _____

```
    default: break;
    }
    drawCurve(curve);
}

void mydisplay() {
    drawCurve(curveNum);
}

void myreshape(int nw, int nh) {
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho2D(0.0, (double)nw, 0.0, (double)nh);
    glClear(GL_COLOR_BUFFER_BIT);
}

void main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(w,h);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Drawing Curves");
    int curveID = glutCreateMenu(drawCurve);
    glutAddMenuEntry("Lemniscate", 1);
    glutAddMenuEntry("Cardioid", 2);
    glutAddMenuEntry("Spiral", 4);
    glutAttachMenu(GLUT_LEFT_BUTTON);
```

Teacher's Signature _____

```
int colorID = glutCreateMenu(colorMenu);
glutAddMenuEntry("Red", 4);
glutAddMenuEntry("Green", 2);
glutAddMenuEntry("Blue", 1);
glutAddMenuEntry("Black", 0);
glutAddMenuEntry("White", 7);
glutAddMenuEntry("Cyan", 3);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawCurve", curveId);
glutAddSubMenu("colors", colorId);
glutAttachMenu(GLUT_LEFT_BUTTON);
myinit();
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);
glutMainLoop();
}
```

Teacher's Signature

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, x[4], y[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;
glColor3f(0.0, 0, 0);
```

Teacher's Signature

```
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)*x[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);
glBegin(GL_POINTS);
glVertex2i(x, 500-y);
glEnd();
glFlush();
flag++;
```

Teacher's Signature

```
}
if(flag>=1 && btn==GLUT_LEFT_BUTTON)
{
glColor3f(0.0,0.0,1.0);
display();
flag=0;
}
}

int main(int argc, char* argv[]){
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowPosition(0,0);
glutCreateWindow("Box");
glutDisplayFunc(display);
glutMouseFunc(mymouse);
myInit();
glutMainLoop();
}
```

Teacher's Signature