Certificate

Name: Ajay Jadhav Class: 7th

Roll No: 1RV18CS400 Exam No:

Institution R.V. College of Engineering.

This is certified to be the bonafide work of the student in the Computer Graphics *Laboratory during the academic year 2020/2021.*

No. of practicals certified _____ *out of* _____ *in the subject of* _____

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Index

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
1.	Bresenham's line drawing Technique	1			
2.	Circle & Ellipse using Bresenham's Circle & ellipse drawing Technique.	5			
3.	3D Sierpinski gasket	8			
4.	Scan-line filling algorithm	10			
5.	House figure with Rotate & Reflect	12			
6.	Cohen-Sutherland line clipping algorithm	14			
7.	Liang-Barsky line Clipping algorithm	17			
8.	Cohen-Hodgeman Polygon Clipping algorithm	20			

Index

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
9.	Car like figure using display list	23			
10.	Color cube & Spin it using opengl	25			
11	Menu with three Entries named Curves, colors & quit.	29.			
12.	Construct Bezier curve.	33			

Date _____

Expt. No. 1 Page No. 1

Program - 1

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

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

int x1, x2, *yc1, y2;
int flag = 0;

void draw_pixel(int x, int y)
{
  glColor3f(1,0,0);
  glbegin(GL_POINTS);
  glvertex2i(x, y);
  glEndFlush();
  glFlush();
}
```

Teacher's Signature _____

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

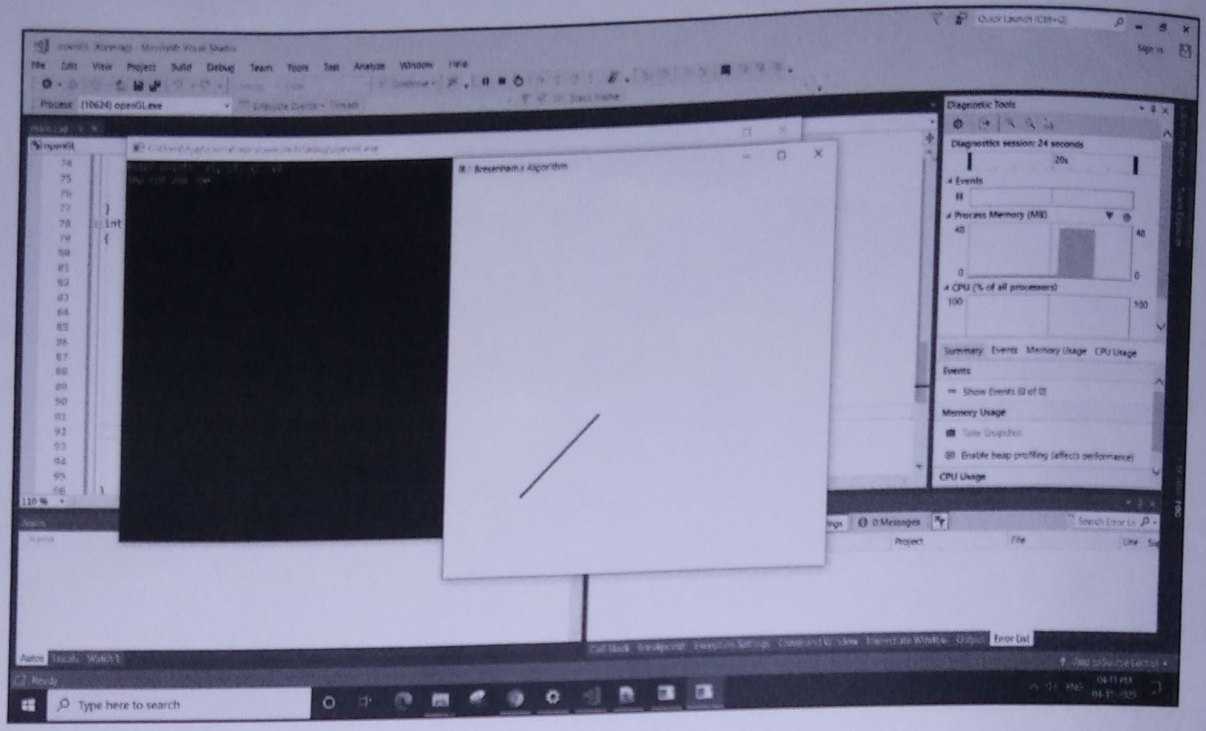
```
else
    e+=inc2;
y+=incy;
    draw_pixel(x,y);
  }
}
glFlush();
}

void init()
{ glClear(GL_COLOR_BUFFER_BIT); glClearColor(1,1,1,1)
  glOrtho2D(-250,250,-250,250);
}

void MyMouse(int button, int State, int x, int y)
{
 Switch(button){
  Case GLUT_LEFT_BUTTON;
   if(State==GLUT_DOWN)
   { if(flag==0)
     { printf("Defining x1, y1"); x1=x-250;
       y1=250-y; flag++; }
     else { printf("x2, y2");
     x2=x-250; y2=250-y; flag=0;
      draw_line();
     }
   } break; } }
```

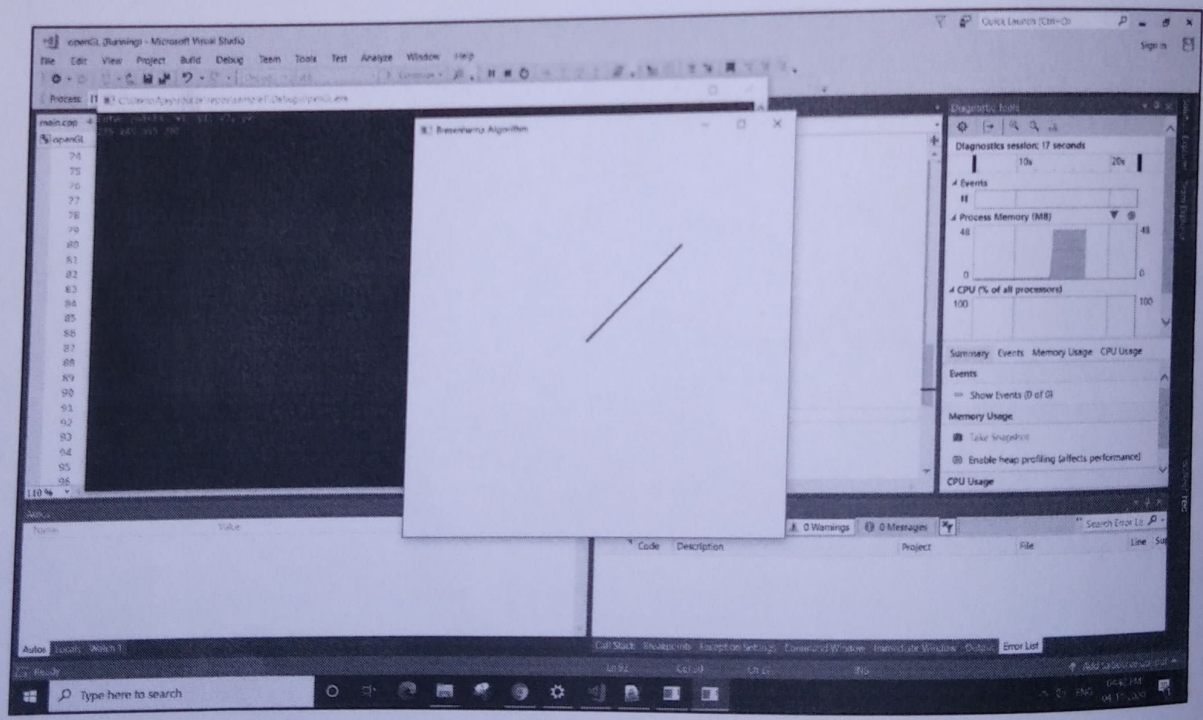
Date

Expt. No. 1 Page No. 4

```
int main( int ac, char* argv[])
{
  cout<< "x1 \n";
  cin>>x1;
  cout<<"y1 \n";   cin>>y1;
  cout<<"x2 \n";   cin>>x2;
  cout<<"y2 \n";   cin>>y2;

  glutInit(&ac, av);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowSize(500,500);
   glutInitWindowPosition(100,200);
    glutCreateWindow("LINE");
  myInit();
  glutMouseFunc(MyMouse);
   glutDisplayFunc(display);
   glutMainLoop()
}
```

OUTPUT :


Bresenham's Algorithm
Type here to search


Bresenham's Algorithm
Type here to search

Expt. No. 2 Date

Page No. 5

Program - 2

write a program to generate a circle & ellipse using Bresenham's circle drawing and ellipse drawing technique. use two windows to draw in one window & ellipse in other window. user can specify i/p through M/k.

```
#include<gl/glut.h>
#include<stdio.h>
#include<Math.h>
#include<windows.h>
int xc, yc, 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)
}

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

Teacher's Signature

```
int P1_x, P2_x, P1_y, P2_y; point1_done = 0;

void MyMouseFunction circle (int button, int state, int x, int y)
{
  if(button == GLUT_LEFT_BUTTON && state = GLUT_DOWN)
  {
    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 Exp: (P2_x - P1_x) * (P2_x - P1_x) + (P2_y - P1_y)
                * (P2_y - P1_y);
    r: (int) Sqrt(Exp));
    circle brea();
    point1_done = 0;
  }
}
```

Expt. No. 2 Date

Page No. 6

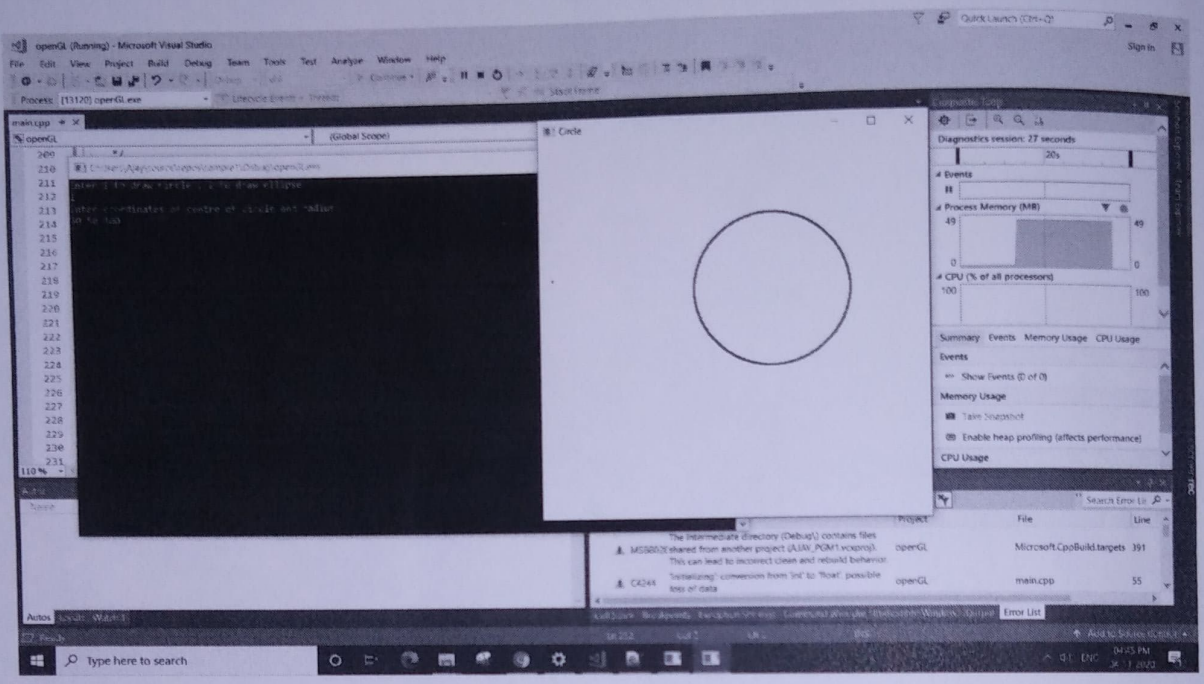
```
//ELLIPSE//
void draw_ellipse(int xce, int yce, int x, int y)
{
  glBegin(GL_POINTS)
  glvertex2i(x+xce, y+yce); glvertex2i(-x+xce, y+yce);
  glvertex2i(x+xce, -y+yce); glvertex2i(-x+xce, -y+yce);
  glEND();
}

void midtellipse()
{ glClear(GL_COLOR_BUFFER_BIT);
  float dx, dy, d1, d2, 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)
  {
    drawellipse(xce, yce, x, y);
    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);
  }
}

  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);
```

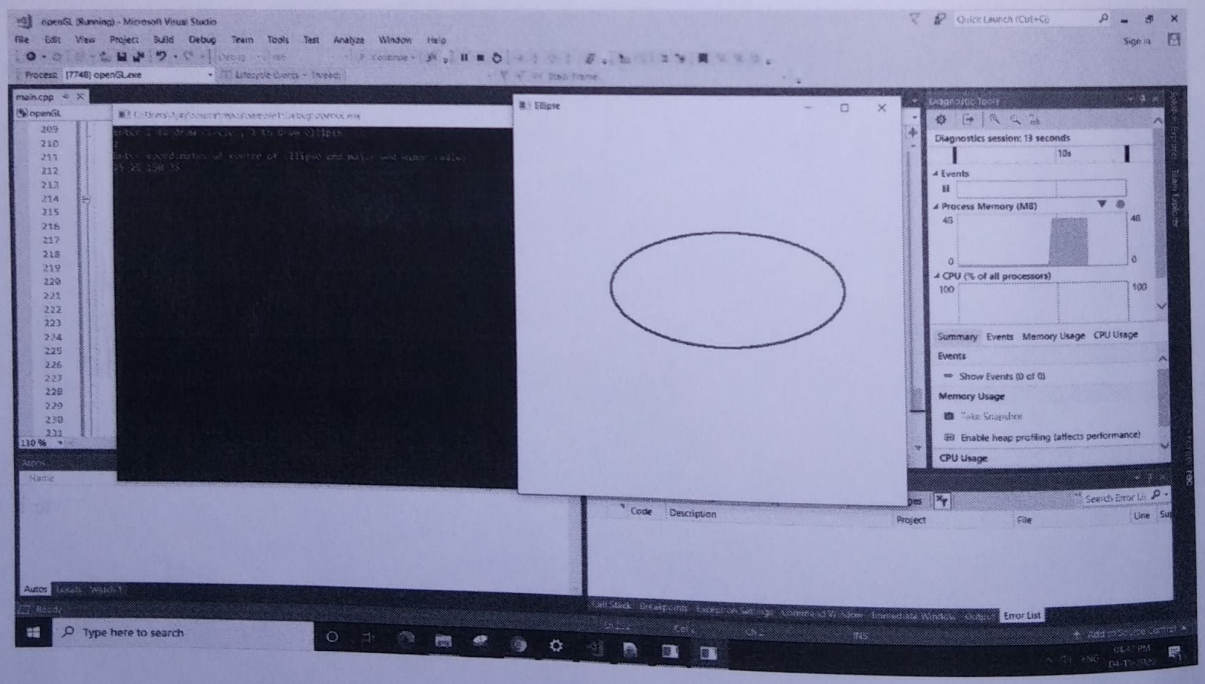
```
if(d2>0)
{ y--;
  dy = dy - (2*rn*rn);
  d2 = d2 + (rn*rn) - dy;
}
else
 {
   y--; x++;
   dn = dn + (2*ry*ry);
   dy = dy - (2*rn+rn);
   d2 = d2 + dn - dy + (rn*rn);
   }
 }
 glFlush();
}

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

```
glutInitWindowSize (500, 500);  //For mouse
glutInitWindowPosition (0, 0);

int id1 = glutCreateWindow ("circle"); glutSetWindow (id1);
glutMouseFunc (myMouseFuncircle); glutDisplayFunc ();
minit ();
glutInitWindowSize (500, 500);
int id2 = glutCreateWindow ("Ellipse");
glutSetWindow (id2);
glutMouseFunc (myMouseFunc);
glutDisplayFunc (Mydrawing);

glutMainLoop ();  //For keyboard
printf ("Enter 1 : circle , 2 : Ellipse\n");
int ch;
Scanf ("%d", &ch);
Switch (ch) {
case 1 : Scanf ("%d %d %d", &xc, &yc, &r);
         glutCreateWindow ('circle');
         glutDisplayFunc (circlebres);
         break;

case 2 : Scanf ("%d %d %d %d", &xce, &yce, &rx, &ry);
         glutCreateWindow ("Ellipse");
         glutDisplayFunc (midtellipse);
         break;
}
minit ()
glutMainloop ();
}
```

Teacher's Signature

OUTPUT :


openGL (Running) - Microsoft Visual Studio
Circle
Diagnostic Tools
Diagnostics session: 27 seconds
Process Memory (MB)
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage
Type here to search


openGL (Running) - Microsoft Visual Studio
Ellipse
Diagnostic Tools
Diagnostics session: 13 seconds
Process Memory (MB)
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
CPU Usage
Type here to search

Date

Expt. No. 3 Page No. 8

Program - 3.

WAP to generate recursively subdivides a tetrahedron to form 3D sierpinski gasket. The no of recursive steps is to be specified at execution time.

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

int n;
typedef float point[3];
point tetra[4]={{0, 100, -100}, {0, 0, 100}, {100, -100, -100},
                {-100, -100, -100}};
void tetrahedron(void); void myint(void);
void divide_triangle(point a, point b, point c, int m)
void draw_triangle(point p1, point p2, point p3);
int main(int argv, char **argc)
{
    printf("Enter no of iterations: ");
    scanf("%d", &n);
    glutInit(&argv, argc);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100, 200);
    glutCreateWindow("Sierpinski Gasket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    myInit();
    glutMainLoop();
}
```

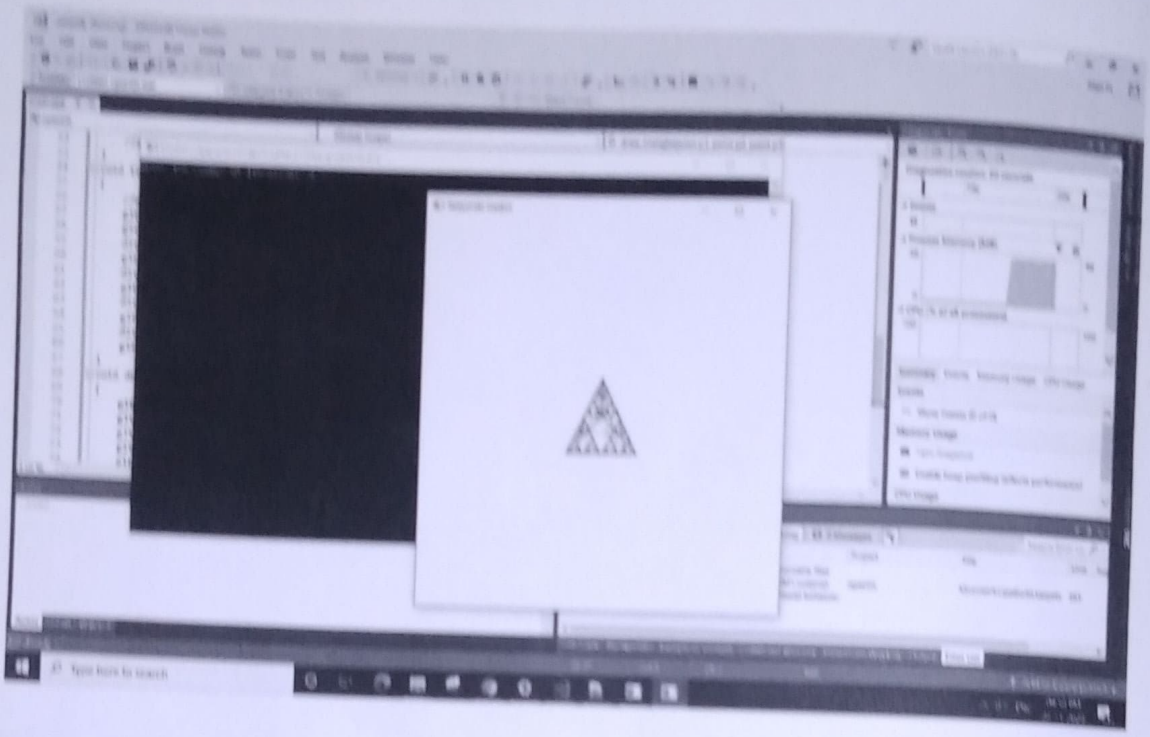
Teacher's Signature

```
void divide_triangle(point a, point b, point c, int m)
{
  Point v1, v2, v3;
  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);
  } else
      draw_triangle(a, b, c);
  void myinit()
{
  glClearColor(1,1,1,1);
  gortho(-500.0, 500.0, -500.0, 500.0, -500.0, 500.0);
}
void tetrahedron(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
```

```
glcolor3f(1.0, 0.0, 0.0);
divide_triangle(tetra[0], tetra[1], tetra[2], n);
glcolor3f(0.0, 1.0, 0.0);
divide_triangle(tetra[3], tetra[2], tetra[1], n);
glcolor3f(0.0, 0.0, 0.0);
divide_triangle(tetra[0], tetra[2], tetra[3], n);
  glFlush();
}

void draw_triangle(Point P1, point P2, point P3)
{
  glBegin(GL_TRIANGLE);
    glvertex3fv(P1);
    glvertex3fv(P2);
     glvertex3fv(P3);
      glEnd();
}.
```

Teacher's Signature

OUTPUT :



Program - 4

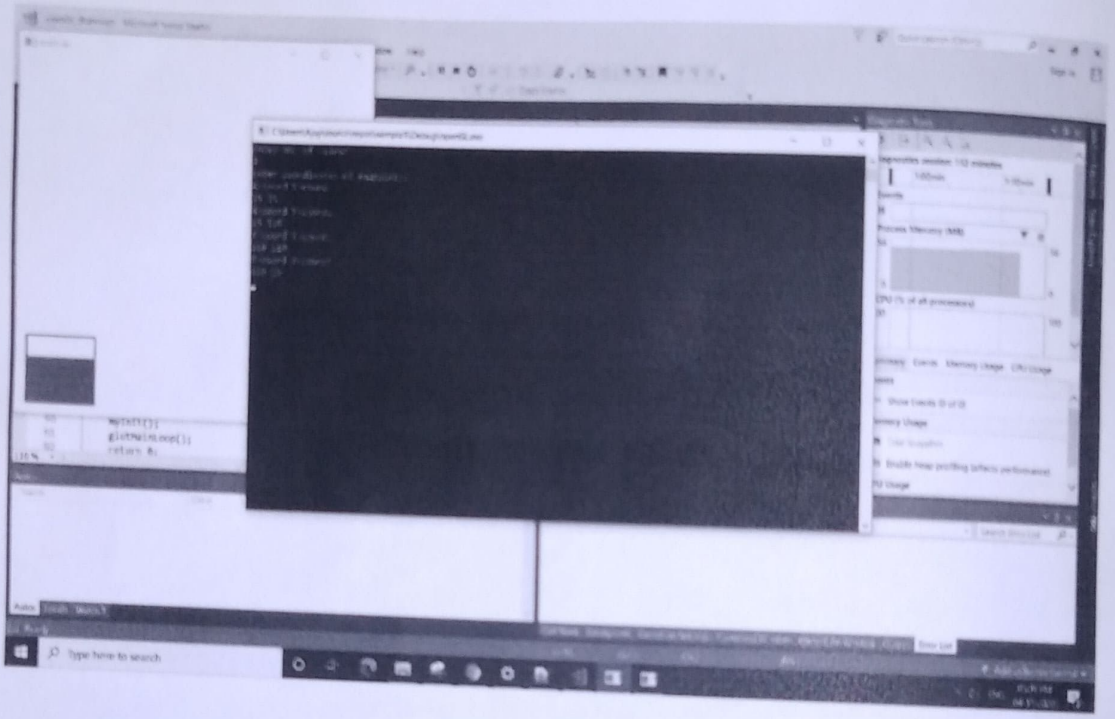
write a program to fill any given Polygon using Scan-line area filling algorithm.

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

using namespace std;
float x[100], y[100];
int n, m, 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)
    int x[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],
        }
        Sort (int x, (int x + m));
        if (m >= 0)
            for (int i = 0; i < m; 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<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 a, char *gg[])
{ glutInit(&ac,av);
  printf("Enter no of sides:\n");
  Scanf("%d",&n);
  printf("Enter coordinates of Endpoints:\n");
  for(int i=0;i<n;i++)
  {
   printf("x-coord Y-coord:\n");
   Scanf("%f %f",&x[i], &Y[i]);
  }
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutWindowSize(500,500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("Scanline");
  glutDisplayFunc(display_filled_polygon);
   myInit();
   glutMainLoop();
}.
```

OUTPUT :



Expt. No. 5 Date

Page No. 12

Program - 5.

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

i) Rotate it about given fixed Point

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

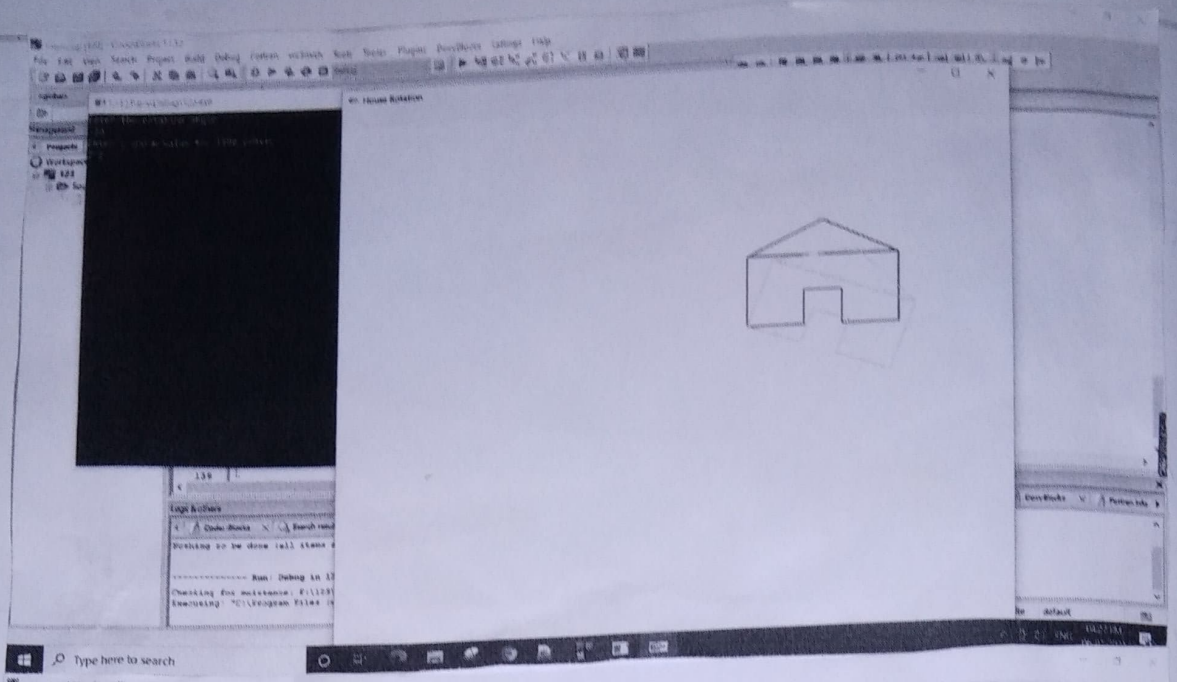
```
#include <gl/glut.h>
#include <math.h>
#include <stdio.h>

float house[11][2] = {{100,200}, {200,250}, {300,200},
{250,150}, {225,100}, {100,200}, {100,100}, {175,100}, {175,150}};
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();
//Normal House
glColor3f(1,0,0); glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
    glVertex2fv(house[i]);
glEnd();
glFlush();
//Rotated.
glPushMatrix();
glTranslatef(100,100,0);
glRotatef(angle,0,0,1);
```

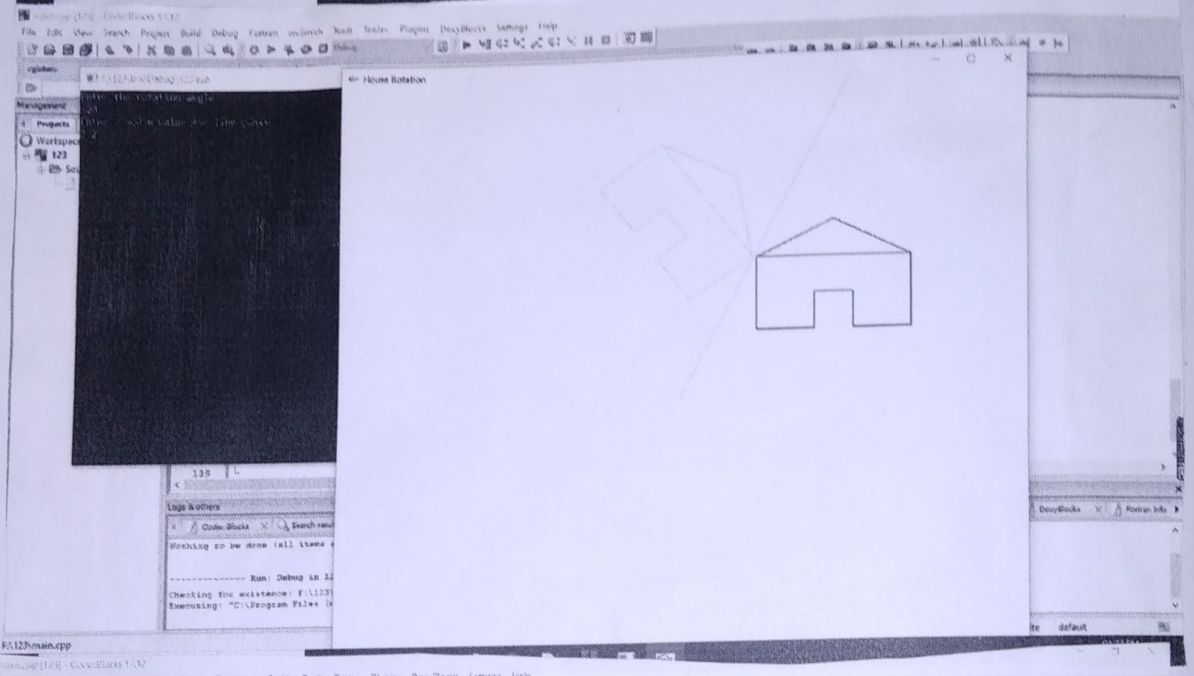
Teacher's Signature

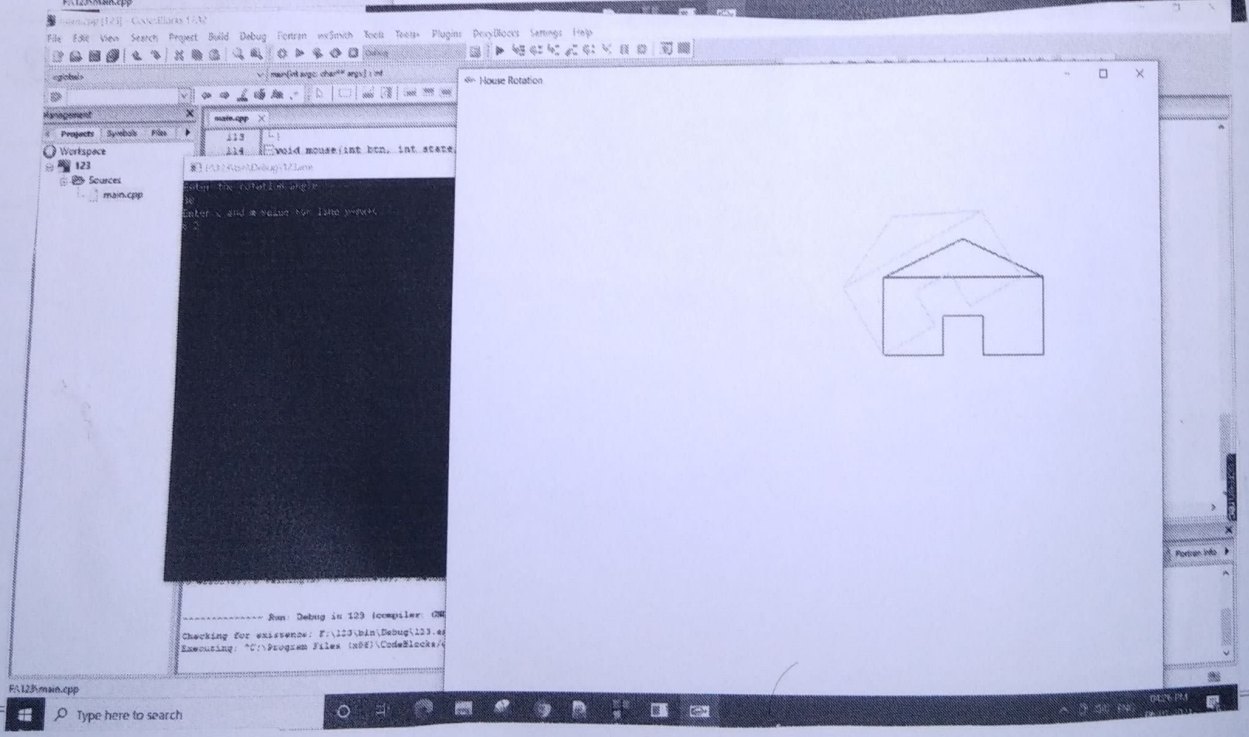
```
glTranslatef(-100,-100,0);
glColor3f(1,1,0);
glBegin(GL_LINE_LOOP);
for(int i=0;i<11;i++)
    glvertex2fv(house[i]);
glEnd();
glPopMatrix(); glFush();
}

void display()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT)GL_DEPTH_BUFFER_BIT);
glMatrinMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-450,450,-450,450);
glMatrinMode(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();
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();

glPushMatrix(); glTranslatef(0,c,0);
theta = atan(m); theta = theta * 180/3.114;
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();
}

void main(int argc, char **argv)
{
printf("Enter the rotation angle\n");
Scanf("%d", &angle);
Printf("Enter c & m value for line y=mx+c\n");
Scanf("%f %f", &c, &m);
glutInit(&argc, argv);
glutInitWindowSize(900,900);
glutInitWindowPosition(100,100);
glutCreateWindow("House Rotation");
glutDisplayFunc(display);
glutMouseFunc(mouse);
myInit();
glutMainLoop();
}
```

Teacher's Signature


House Rotation
Type here to search


House Rotation
F:\123\main.cpp


House Rotation
Type here to search

Date

Expt. No. 6 Page No. 14

Program - 6.

write a program to implement the cohen-Sutherland line clipping algo. Make provision to specify the input for multiple lines, window for clipping & 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=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|=LEFT;
  else if (x>xmax) code|=RIGHT;
  return code; }
```

Teacher's Signature

```
void CohenSuther(double x0, double y0, double x1, double y1)
{
Outcode outcode0, outcode1, outcode;
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);
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);
```

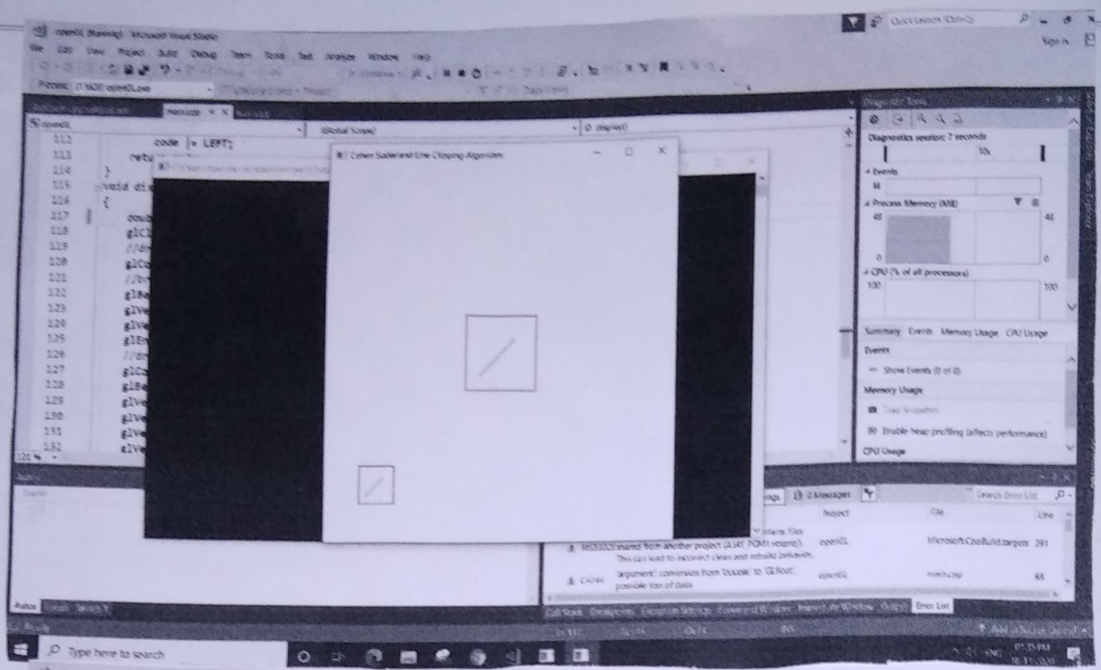
```
glvertex2f(xmin, ymin);
glvertex2f(xmax, ymin);
glvertex2f(xmax, ymax);
glvertex2f(xmin, ymax); glEnd();
for(int i=0; i<n; i++)
{
  glBegin(GL_LINES);
  glvertex2d(ds[i].x1, ds[i].y1);
  glvertex2d(ds[i].x2, ds[i].y2);
  glEnd();
}
for(int i=0; i<n; i++)
   CohenSuther(ds[i].x1, ds[i].y1, ds[i].x2, ds[i].y2);
glFlush();
}

void myinit()
{
glclearcolor(1,1,1,1);
glcolor3f(1,0,0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluortho2D(0,500,0,500);
}
```

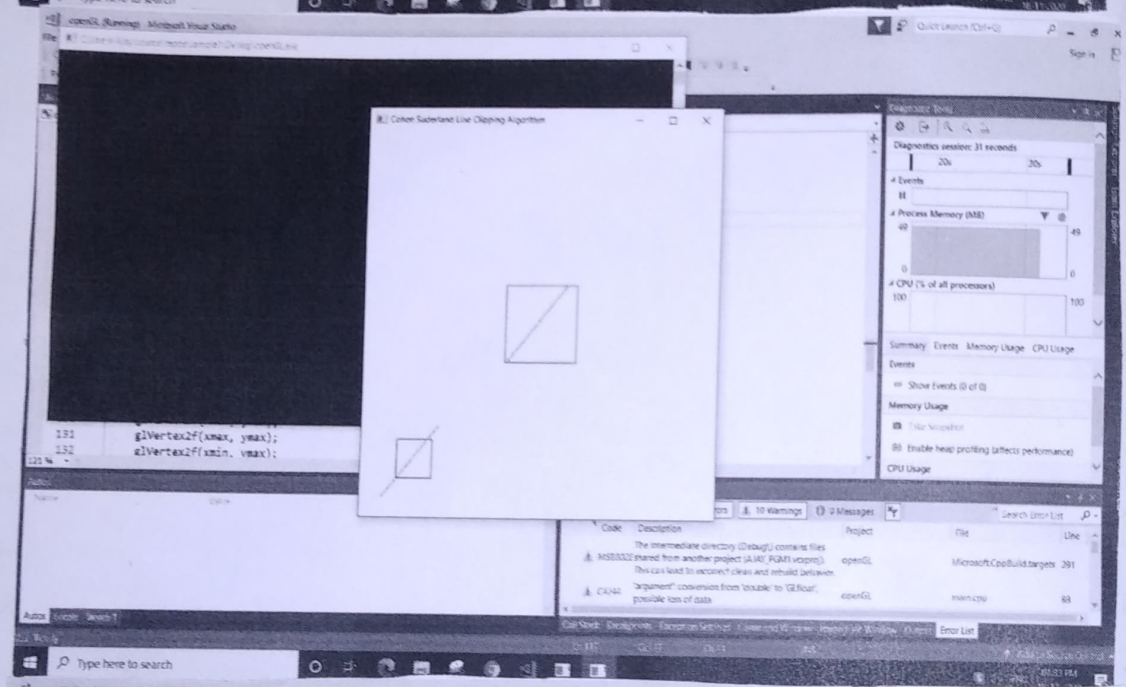
Date

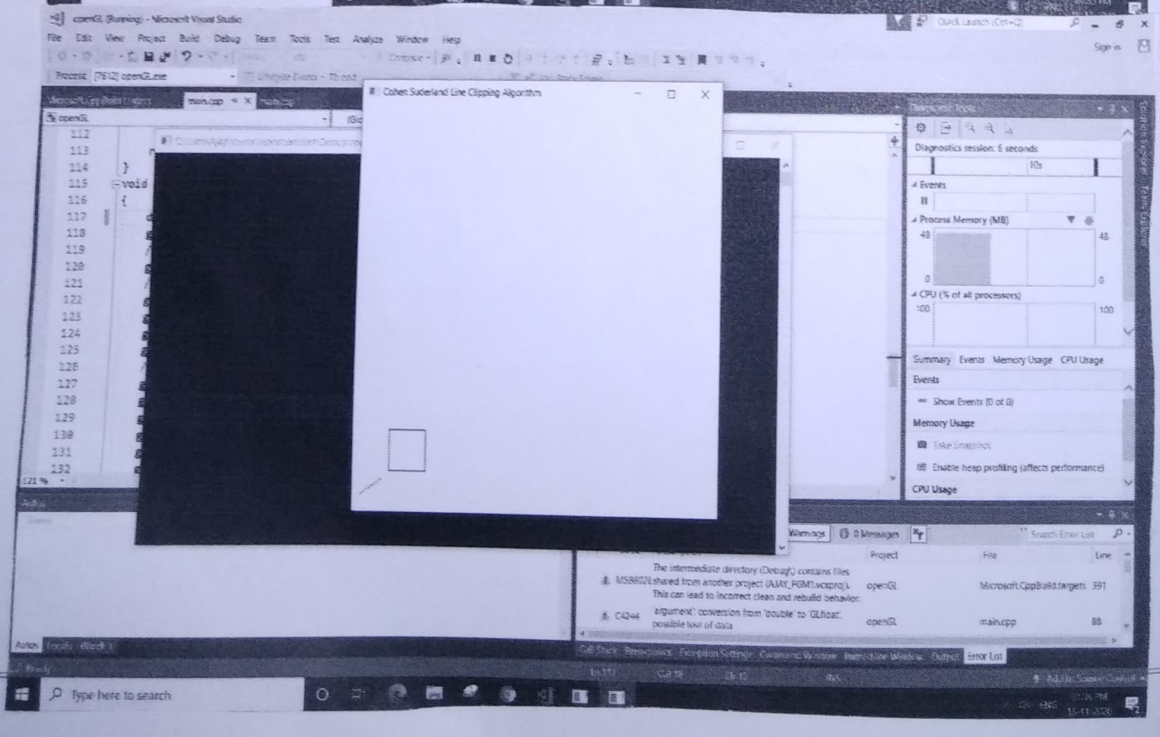
Expt. No. 6 Page No. 16

```
void main(int argc, char argv)
{
 printf("Enter window coordinates : \n");
 scanf("%lf %lf %lf %lf", &xmin, &ymin, &xmax, &ymax);
 printf("Enter viewport coordinates \n");
 scanf("%lf %f %lf %f", &xvmin, &yvmin, &xvmax, &yvmax
 printf("Enter no of lines : \n");
 scanf("%d", &n);
 for(int i=0; i<n; i++)
 { printf("Enter line endpoints : \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(500, 500);
   glutInitWindowPosition(0, 0);
   glutCreateWindow("clip");
   myinit();
   glutDisplayFunc(display);
   glutMainLoop();
}
```

Teacher's Signature


Cohen Sutherland Line Clipping Algorithm


Cohen Sutherland Line Clipping Algorithm


Cohen Sutherland Line Clipping Algorithm

Program - 7.

write a program to Implement the Liang - Barsky line clipping algorithm. Make Provision to specify the i/p for multiple lines, window for Clipping & viewport for displaying the clipped image.

```
#include <stdio.h>
#include <GL/glut.h>
double xmin, ymin, xmax, ymax, xvmin, yvmin, xvmax, yvmax;
struct line_segment {
    int x1, y1, x2, 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);
}
```

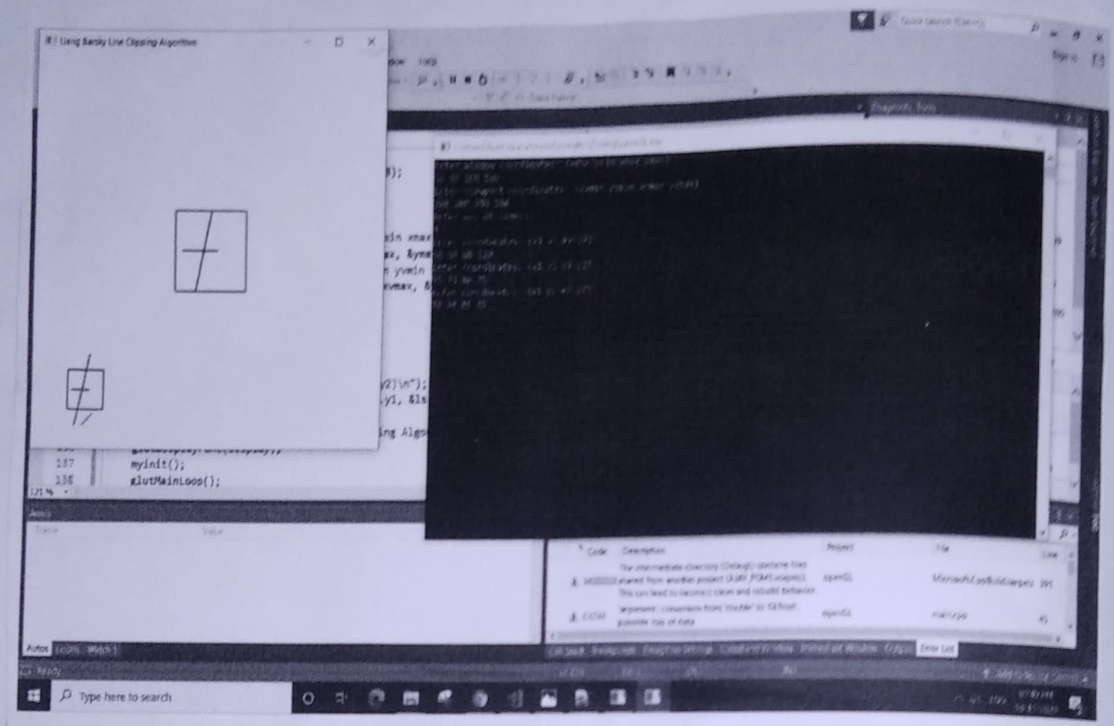
```
void LiangBarskyLineClipAndDraw (double x0, double y0,
                                 double x1, double y1)
{
 double dx=x1-x0, dy=y1-y0, u1=0.0, u2=1.0;
 glColor3f(1.0, 0.0, 0.0);
 glBegin(GL_LINE_LOOP);
 glVertex2f(xvmin, yvmin);
 glVertex2f(xvmax, yvmin);
 glVertex2f(xvmax, yvmax);
 glVertex2f(xvmin, 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;
          }
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(0.0, 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();
}

glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xmin, ymin);
glVertex2f(xmax, ymin);
glVertex2f(xmax, ymax);
glVertex2f(xmin, ymax);
glEnd();

for(int i=0; i<n; i++)
  LiangBarskyLineClipAndDraw(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);
 glLineWidth(2.0);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluOrtho2D(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("%lf%lf%lf", &xmin, &ymin, &xmax, &ymax);
Printf("Enter viewport coordinates: \n");
Scanf("%lf%lf%lf%lf", &xvmin, &yvmin, &xvmax
                                    & yvmax);
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("Liang Barsky Algo");
 glutDisplayFunc(display);
 myinit(); glutMainLoop();
}
```

Teacher's Signature

OUTPUT :


Liang Barsky Line Clipping Algorithm
myinit();
glutMainLoop();
Type here to search

Date _____

Expt. No. 8 Page No. 20

Program – 8.

write a program to implement the cohen. Hodgeman polygon clipping algorithm. Make provision to specify the input polygon & window for clipping.

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
int poly_size, poly_points[20][20], 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;
}
```

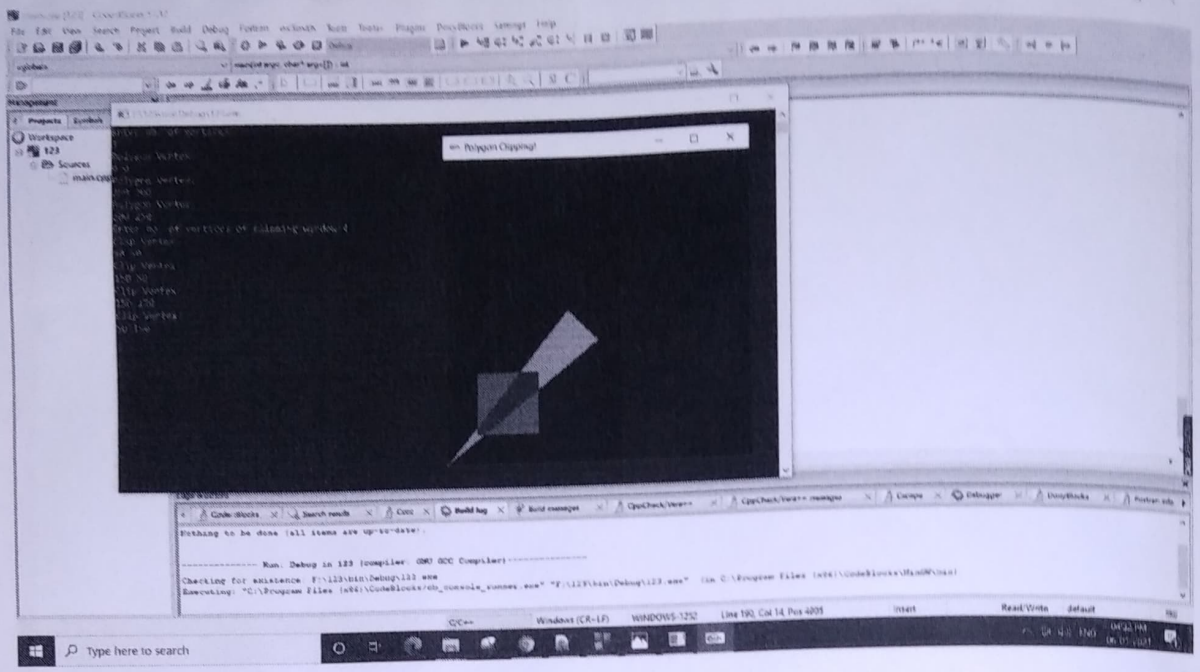
Teacher's Signature _____

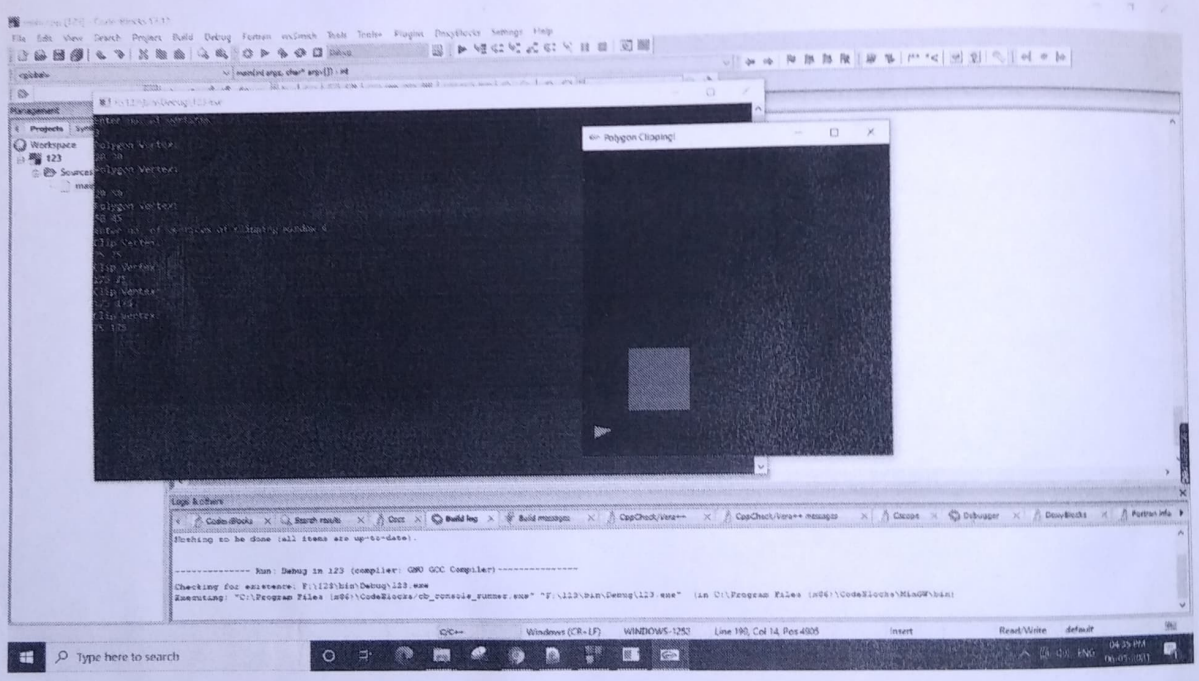
```
void clip(int poly_points[][2], int &poly_size,
          int x1, int y1, int x2, int y2)
{
  int new_points[MAX_POINTS][2], new_poly_size = 0;
  for(int i = 0; i < poly_size; i++)
  {
    int k = (i+1) % poly_size;
    int ix = poly_points[i][0], iy = poly_points[i][1];
    int kx = poly_points[k][0],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

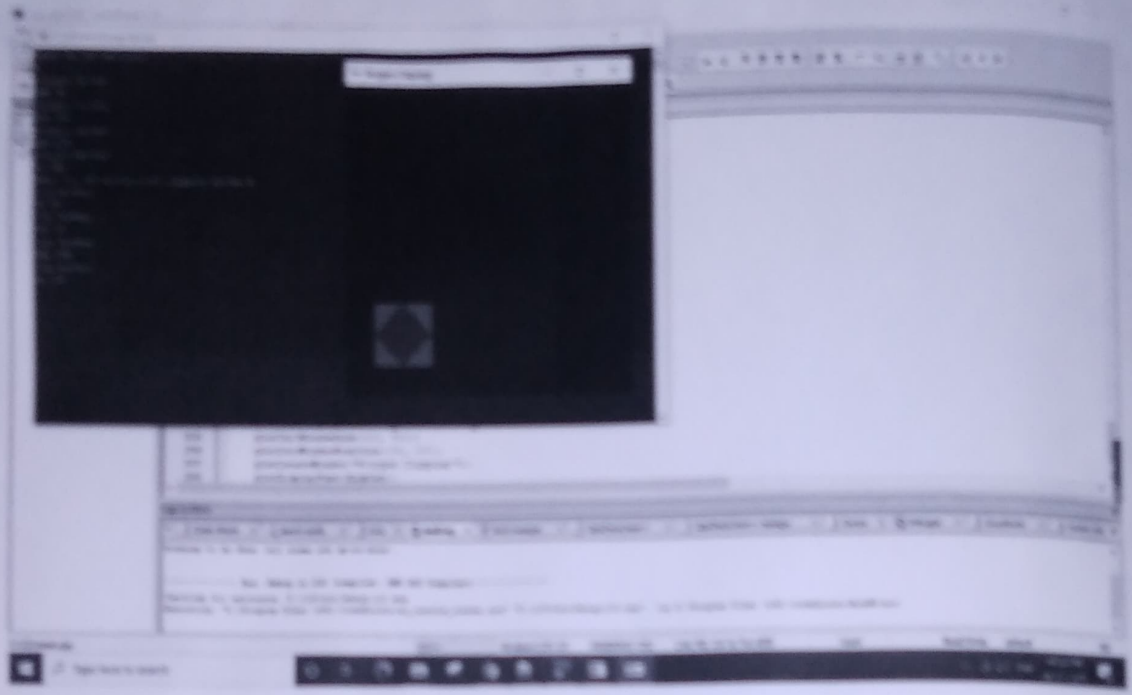
```
            new_points[new_poly_size][1] = y_intersect(x1,
                y1, x2, y2, ix, iy, kx, ky);
            new_poly_size++;
        }

    else
    {
    }
}
poly_size = new_poly_size;
for(int i=0; i<poly_size; i++)
{
    poly_points[i][0] = new_points[i][0];
    poly_points[i][1] = new_points[i][1];
} }

void init() {
    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 display()
{
    init();
    glColor3f(1.0f, 0.0f, 0.0f);
    drawPoly(clippes_points, clipper_size);
```

```
glColor3fC 0.0f, 1.0f, 0.0f);
drawpoly (org-poly-points, org-poly-size);

for(iAt = 0; i < clipper-size; i++)
{ int k = (i+1) % clipper_size;
   Cli> (poly-points, Poly-Size, clipper_points [i][0],
         clipper-points[i][1], clipper_points[k][0]);
} glColor3f (0.0f, 0.0f, 1.0f);
   drawPoly (Poly-points, poly-Size); glFlush();
}
int main (int argc, char * argv[])
{ printf (" Enter no of vertices : \n");
  scan_s (" %d", &poly-size);
  org-poly-size = poly-size;
  for(int i=0; i<poly-size; i++)
  {
    scanf_s (" %d %d", &Poly-Points[i][0], &Poly-Points[i][
    org-Poly-Points[i][0] : Poly-Points[i][0];
    org-Poly-Points[i][1] : Poly-Points[i][1];
  }
printf (" Enter no of vertices of clipping window");
scanf (" %d", &clipper-size);
for( int i=0 ; i < clipper-size; i++)
{ scanf_s (" %d %d", &clipper_points[i][0], &clipper-points[i][
} glutInit (&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowPosition (100, 100);
glutCreateWindow (" Polygon Clipping !");
glutDisplayFunc (display); glutMainLoop();
return 0;
}
```

Teacher's Signature


Polygon Clipping!
Nothing to be done (all items are up-to-date).
Run: Debug in 123 (compiler: GNU GCC Compiler)
Checking for existence: F:\123\bin\Debug\123.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "F:\123\bin\Debug\123.exe"
Type here to search


Polygon Clipping!
Nothing to be done (all items are up-to-date).
Run: Debug in 123 (compiler: GNU GCC Compiler)
Checking for existence: F:\123\bin\Debug\123.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "F:\123\bin\Debug\123.exe"
Type here to search



Date

Expt. No. 9

Page No. 23

Program - 9.

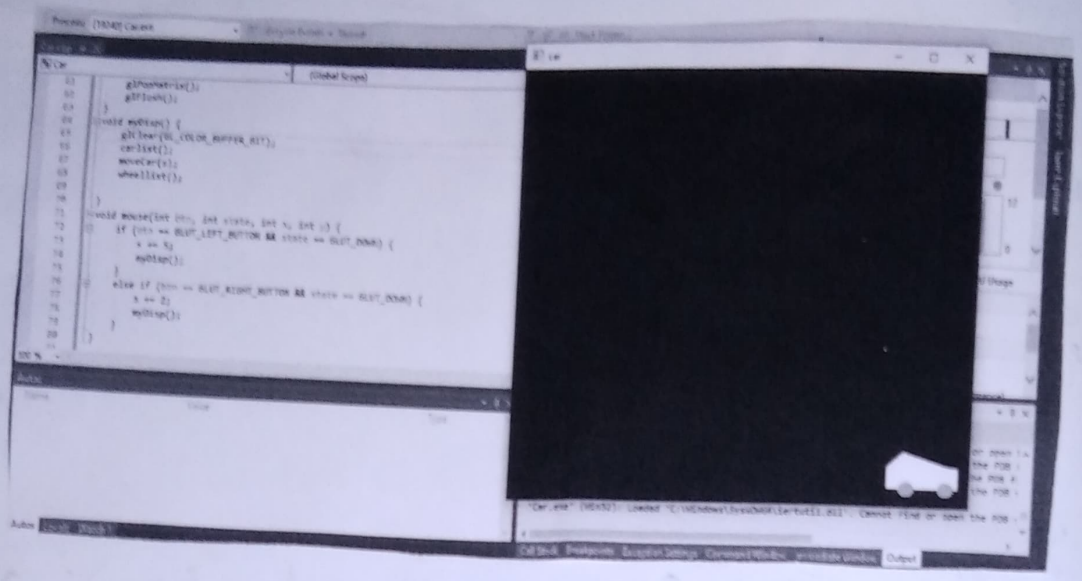
write a program to model a car like figure using display lists & move a car from one to other end of screen. user is able to control the speed with mouse.

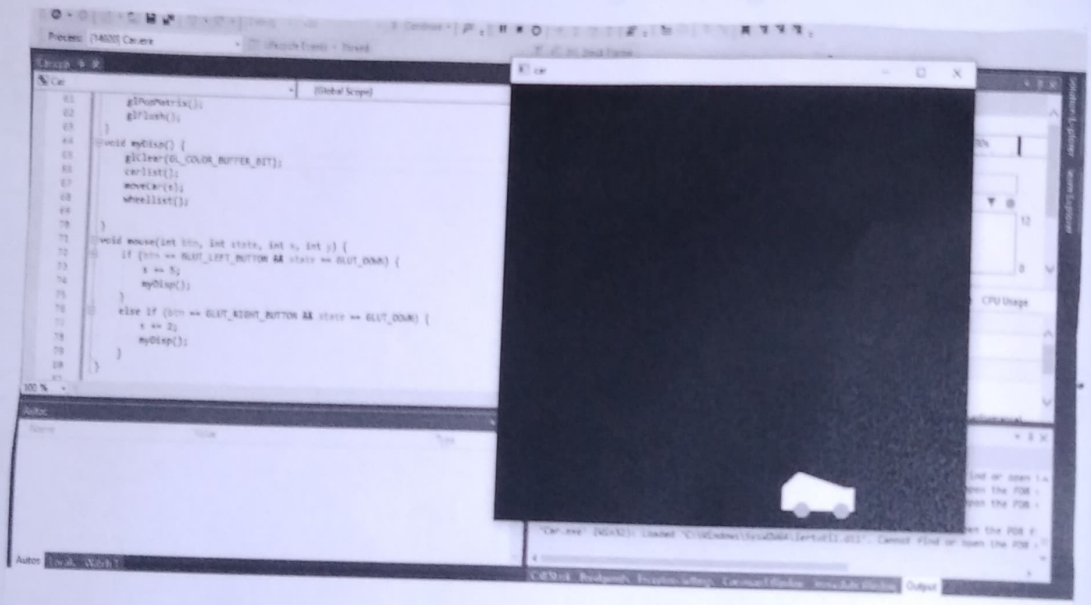
```
#include <GL/glut.h>
#include < math.h>
#include < stdio.h>
#define CAR 1
#define WHEEL 2
float s=1;
void carlist(){
    glNewList(CAR, GL_COMPILE);
    glColor3f(1,1,1);
    glBegin(GL_POLYGON);
    glVertex3f(0,25,0)
    glVertex3f(90, 25, 0);
    glVertex3f(90, 55, 0);
    glVertex3f(80, 55, 0)
    glVertex3f(20, 75, 0);
    glVertex3f(0, 55, 0);
     glEnd(); glEndList();
}
void wheellist(){
    glNewList(WHEEL, GL_COMPILE_AND_EXECUTE);
    glColor3f(0,1,1);
    glutSolidSphere(10, 25, 25);
    glEnd(); glEndList();
}
```

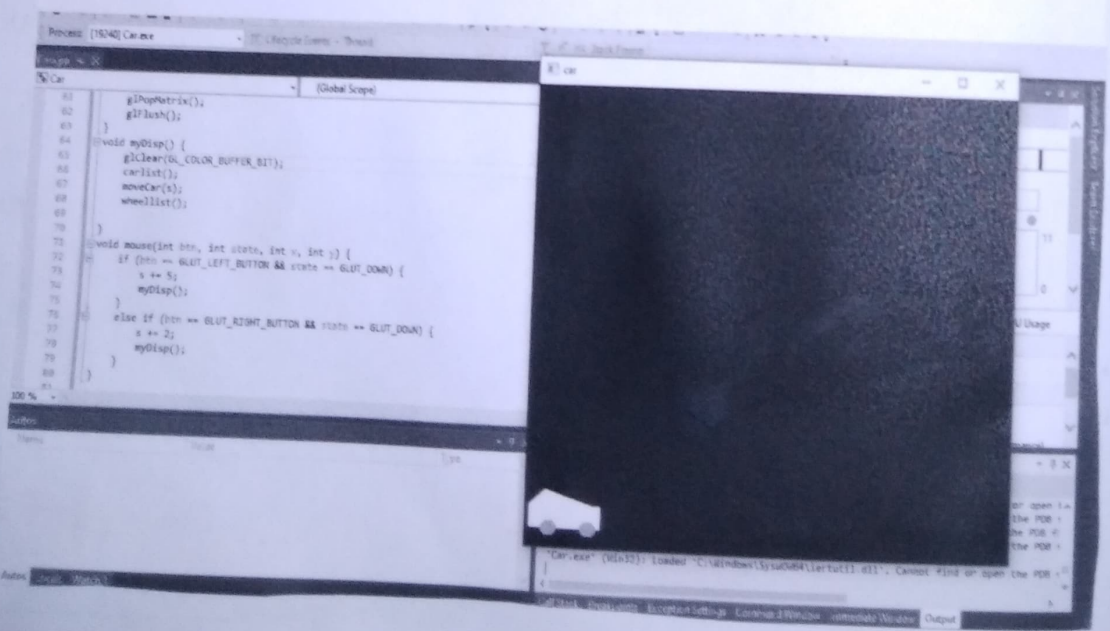
Teacher's Signature

```
void mykyboard (unsigned char Ky, int x, int y)
{
 Switch (Ky) {
 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);
}
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(); }

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);
  glutInitWindowSize(600, 500);
  glutInitWindowPosition(100, 100);
  glutCreateWindow("Car");
  myInt();
  glutDisplayFunc(myDisp)
  glutMouseFunc(mouse);
  glutKeyboardFunc(myKeyboard);
  glutMainLoop();
}
```


car
glPopMatrix();
glFlush();
}
void myDisp() {
glClear(GL_COLOR_BUFFER_BIT);
carlist();
moveCar(x);
wheellist();
}
void mouse(int btn, int state, int x, int y) {
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
x += 5;
myDisp();
}
else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
x += 2;
myDisp();
}
}


car
glPopMatrix();
glFlush();
}
void myDisp() {
glClear(GL_COLOR_BUFFER_BIT);
carlist();
moveCar(x);
wheellist();
}
void mouse(int btn, int state, int x, int y) {
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
x += 5;
myDisp();
}
else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
x += 2;
myDisp();
}
}


car
glPopMatrix();
glFlush();
}
void myDisp() {
glClear(GL_COLOR_BUFFER_BIT);
carlist();
moveCar(x);
wheellist();
}
void mouse(int btn, int state, int x, int y) {
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
x += 5;
myDisp();
}
else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
x += 2;
myDisp();
}
}

Expt. No. 10 Date

Page No. 25

Program - 10

Write a program to create a color cube & spin it using openGL transformations.

```
#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, -1.0,1.0,1.0};
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};
GLfloat
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};
```

Teacher's Signature

```
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_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
                 cube Indices);
  glBegin(GL_LINES);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(1.0, 1.0, 1.0);
  glEnd();
  glFlush();
}
```

Teacher's Signature

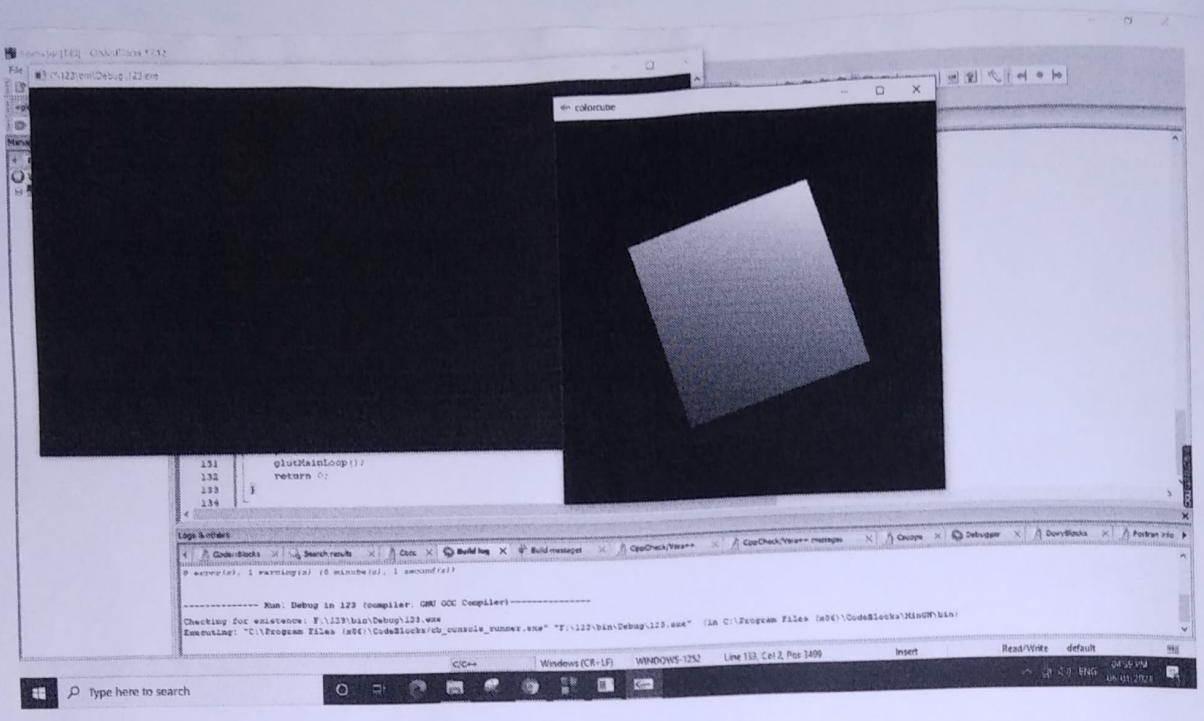
```
void spinCube()
{
  delay(0.01);
  theta[axis] += 2.0;
  if(theta[axis] > 360.0) theta[axis] -= 360.0;
  glutPostRedisplay();
}
void mouse(int btn, int state, int x, int y)
{
  if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
  if(btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 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.
  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);
}
```

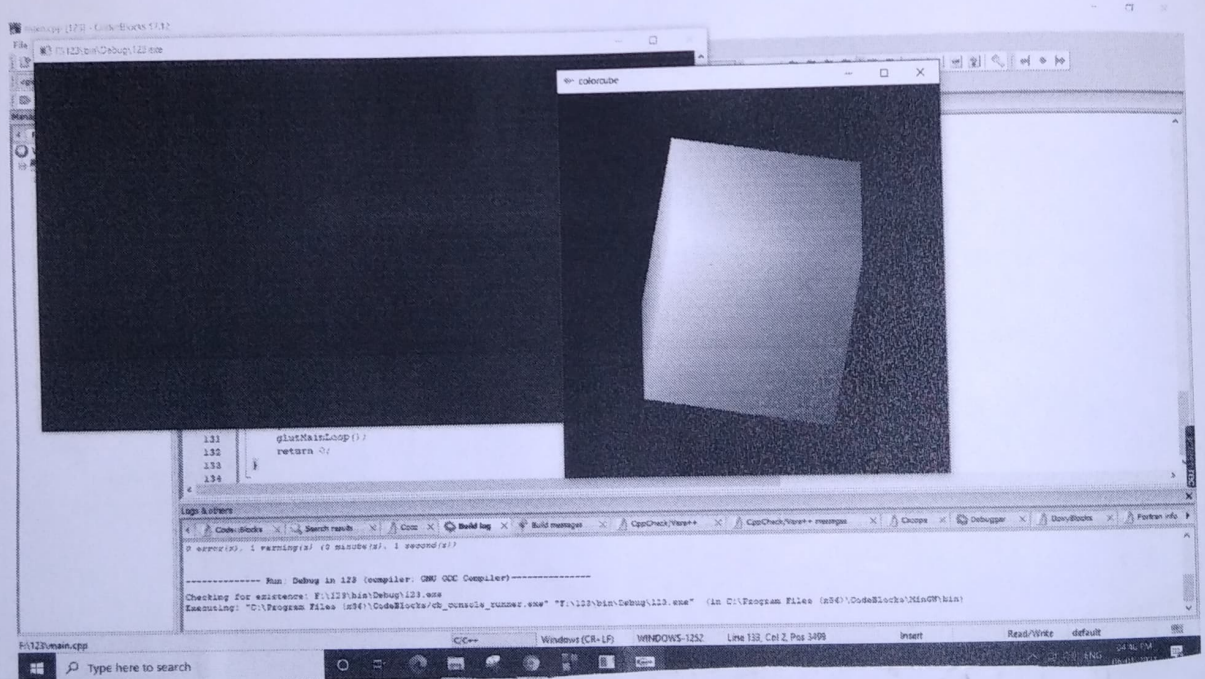
Date

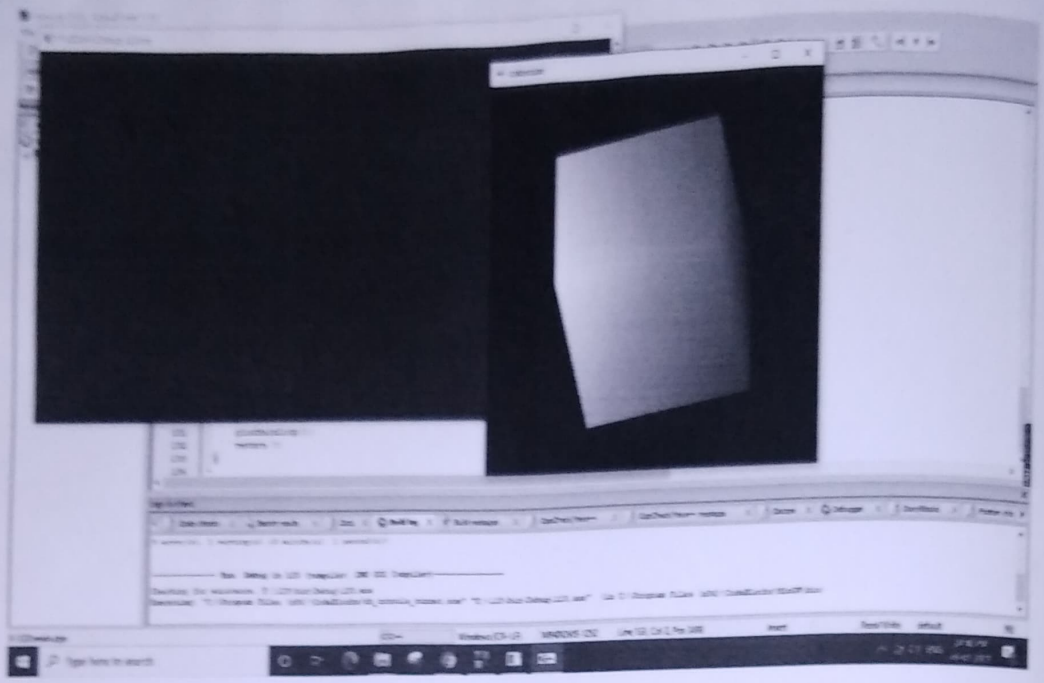
Expt. No. 10 Page No. 27

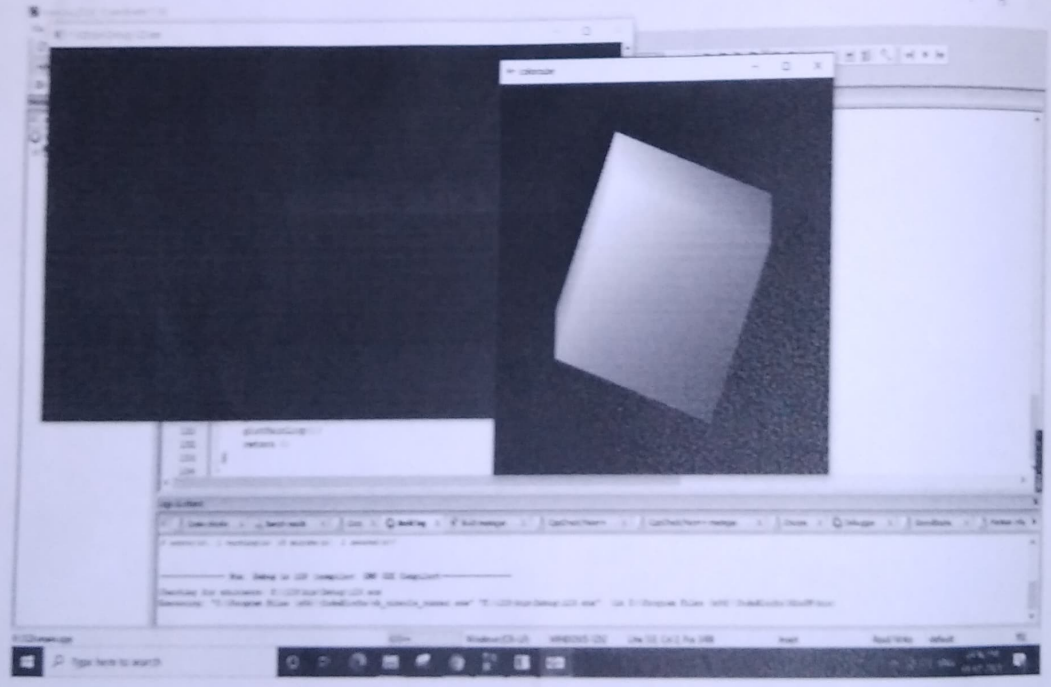
```
void main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition(100, 100);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Color cube");
  glutReshapeFunc(myReshape);
  glutDisplayFunc(displaySingle);
  glutIdleMouseFunc(mouse);
  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();
}
```

Teacher's Signature


colorcube
glutMainLoop();
return 0;
}
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
0 error(s), 1 warning(s) (0 minute(s), 1 second(s))
---------- Run: Debug in 123 (compiler: GNU GCC Compiler)----------
Checking for existence: F:\123\bin\Debug\123.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "F:\123\bin\Debug\123.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 133, Col 2, Pos 3409
Insert
Read/Write
default
Type here to search


colorcube
glutMainLoop();
return 0;
}
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
0 error(s), 1 warning(s) (0 minute(s), 1 second(s))
---------- Run: Debug in 123 (compiler: GNU GCC Compiler)----------
Checking for existence: F:\123\bin\Debug\123.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "F:\123\bin\Debug\123.exe" (in C:\Program Files (x86)\CodeBlocks\MinGW\bin)
F:\123\main.cpp
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 133, Col 2, Pos 3498
Insert
Read/Write
default
Type here to search





Date

Expt. No. 11 Page No. 29

Program - 11.

Create a menu with 3 entries named curves, colors & quit. The Entry Curves has a Sub menu which has 4 entries. The color menu has Sub menu with all 8 colors of RGB color model. WAP to creat the above hierarchical menu & attach appropriate services to each menu Entries.

```
#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 myint (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 ();
}
```

Teacher's Signature

```
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;
  dfault: break
}
  theta= dtheta;
  while(theta<twoPi){
    Switch(CurveNum){
    Case limacon: r=a*cos(theta)+b: breaker;
    Case cardioid: r=a*(1+cos(theta)); break;
    Case Spiral: r=(a/4.0)*theta; break;
    dfault: break;
}
    CurvePt[1].x= x0+r*cos(theta);
    CurvePt[1].y= y0+r*Sin(theta);
    lineSegment(CurvePt[0], CurvePt[1]);
```

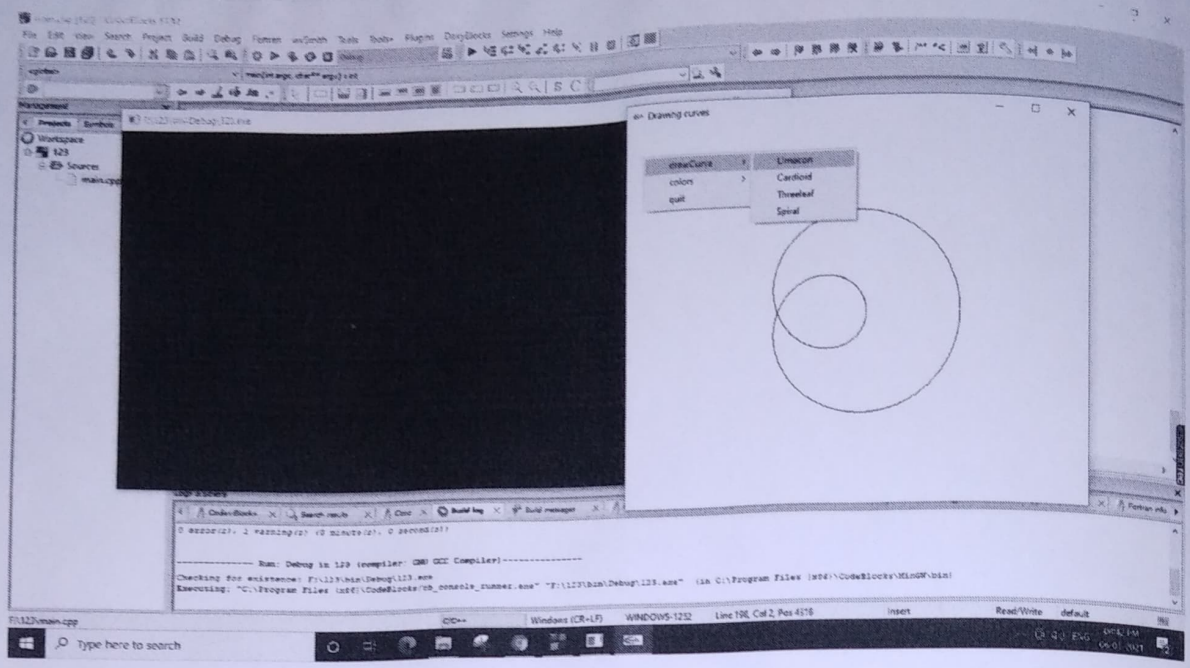
```
        curvePt[0].x = curvePt[1].x;
        curvePt[0].y = curvePt[1].y;
        theta += dtheta;
    }
}.
void ColorMenu(int id) {
    switch(id) {
    case 0: break;
    case 1: red=0; green=0; blue=1; break;
    case 2: red=0; green=1; blue=0; break;
    case 3: red=0; green=1; blue=1; break;
    case 4: red=1; green=0; blue=0; 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;
    }
}
```

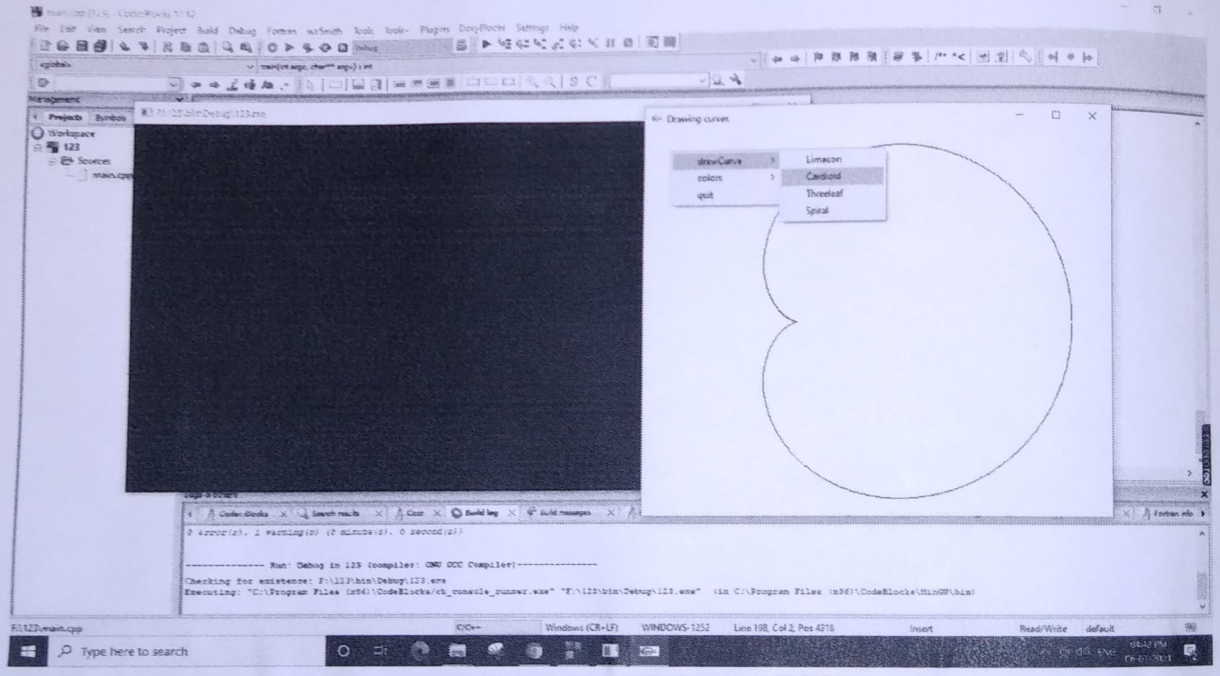
```
void mydisplay(){
    printf("1 - limacon\n2 - cardioid\n3 - threeleaf
        \n4 - Spiral\n");
    Scanf_s("%d", &curveNum);
}

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 main(int argc, char ** argv){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(w, h);
    glutCreateWindowPosition(100, 100);
    glutCreateWindow("Drawing curves");
    int curveId = glutCreateMenu(drawCurve);
    glutAddMenuEntry("Limacon", 1);
    glutAddMenuEntry("Cardioid", 2);
    glutAddMenuEntry("Threeleaf", 3);
    glutAddMenuEntry("Spiral", 4);
    glutAttachMenu(GLUT_LEFT_BUTTON);
```

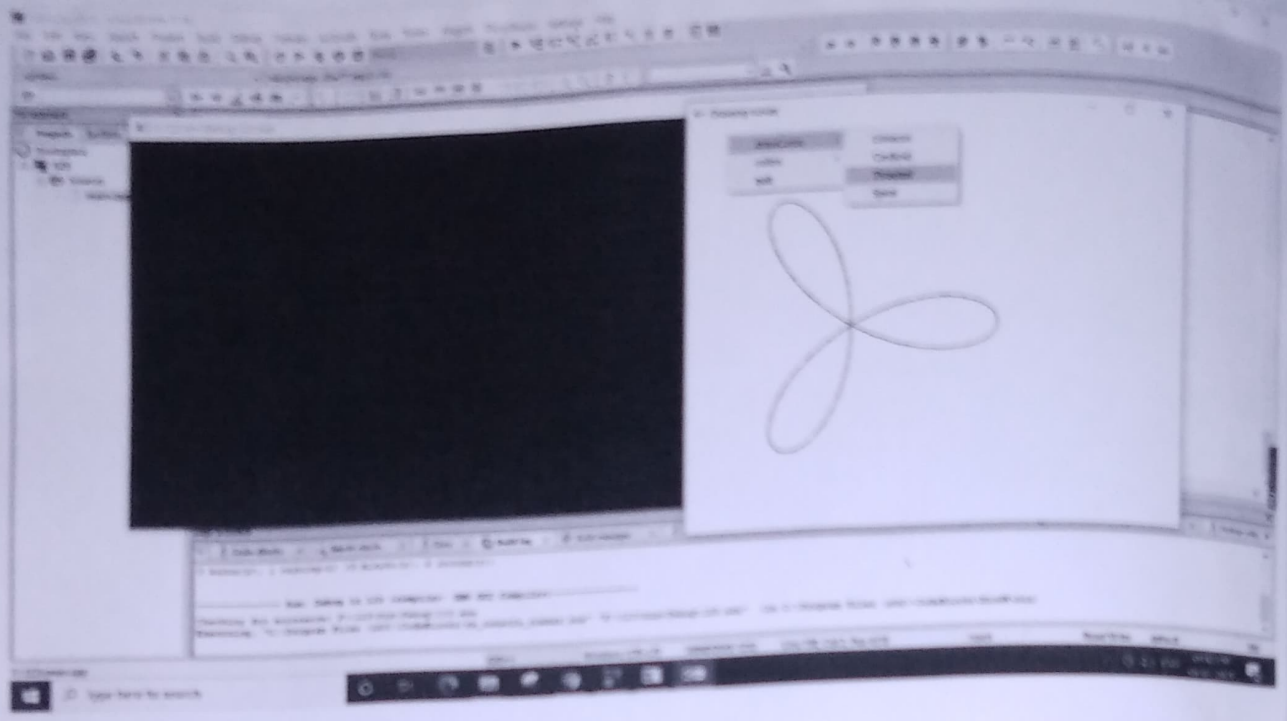
Teacher's Signature

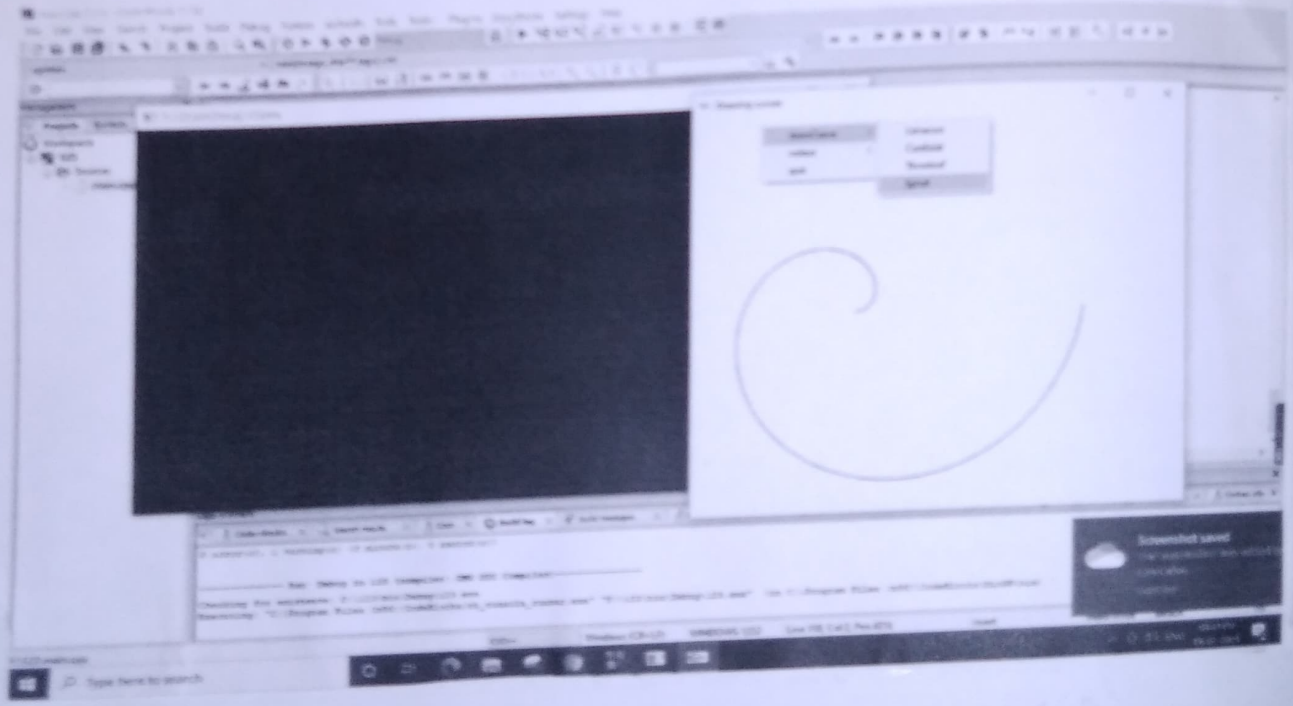
```
int colorId: glutCreateMenu(colorMenu);
glutAddMenuEntry("Red" 4);
glutAddMenuEntry("Green" 2);
glutAddMenuEntry("Blue", 1);
glutAddMenuEntry("Black", 0);
glutAddMenuEntry("Yellow", 6);
glutAddMenuEntry("Cyan", 3);
glutAddMenuEntry("Meogenta", 5);
glutAddMenuEntry("white", 7);

glutAttachMenu(GLUT-LEFT-BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawcurve", curveId);
glutAddSubMenu("colors", colorId);
glutAddMenuEntry("quit", 3);
glutAttachment(GLUT_LEFT-BUTTON);
myinit();
glutDisplayFunc(mydisplay);
glutReshapeFunc(myreshape);

glutMainloop();
}
```

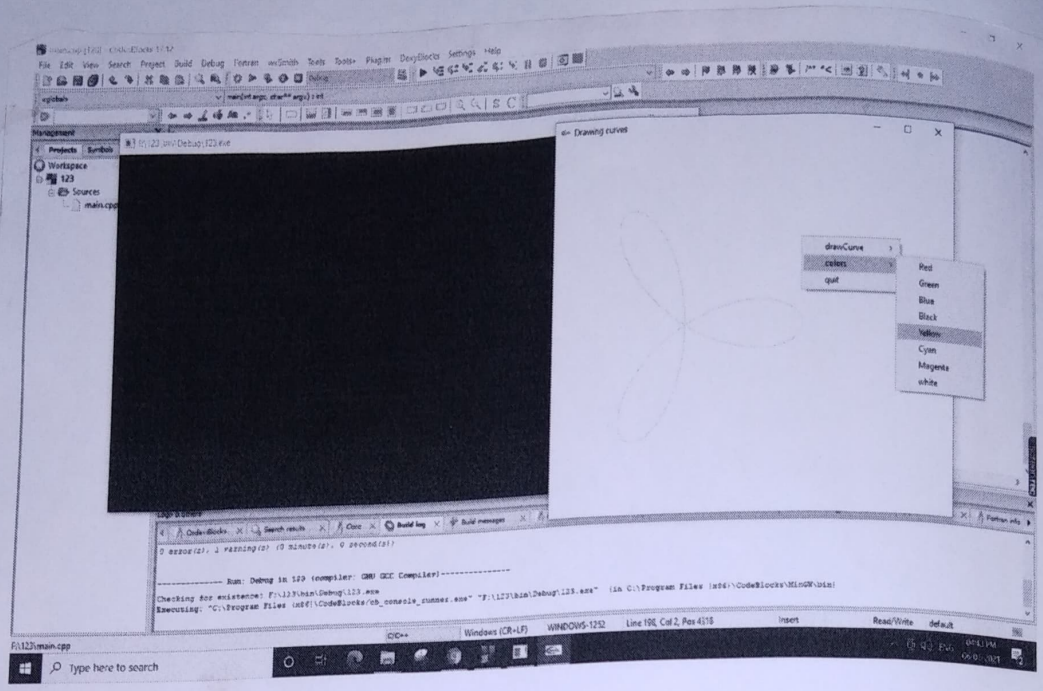
Teacher's Signature

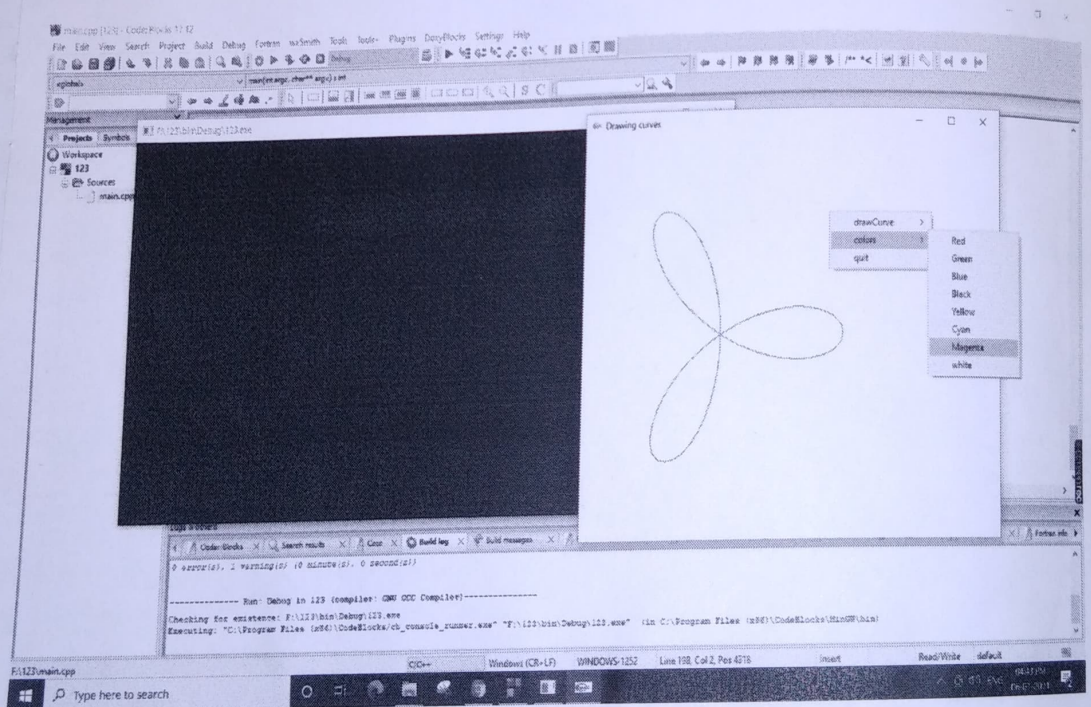

Drawing curves
drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral
Type here to search


Drawing curves
drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral
Type here to search






Drawing curves
drawCurve
colors
quit
Red
Green
Blue
Black
Yellow
Cyan
Magenta
white
Type here to search


Drawing curves
drawCurve
colors
quit
Red
Green
Blue
Black
Yellow
Cyan
Magenta
white
Type here to search

Program - 12.

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, xi[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) * 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)
  {
```

```
xi[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);
  display();
  flag = 0;
}
}

int main(int argc, char * argv[]) {
    glutInit(&argc, argv);
{
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutInitWindowPosition(0, 0);
  glutCreateWindow("B2");
  glutDisplayFunc(display); glutMouseFunc(mymouse)
  myInit();
} glutMainLoop();
```

Teacher's Signature