CERTIFICATE

Date: 06 / 01 /2021

This is to certify that ~~Mr./~~ Ms. RAMYASHREE.V

of Class 7th Division C Roll No. 1RV18CS427

has satisfactorily completed the course experiments in practical

COMPUTER GRAPHICS (CG)

in the academic year of 2020 / 2021

in the Institution R.V.C.E.

Teacher Examiner Principal

Institution Rubber Stamp

Experiment Name / No.: 01 - Bresenham's line drawing

Camlin Page No. 1
Date / /

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

```
#include <iostream>
#include <GL/glut.h>
#include <time.h>
using namespace std;
int x1, x2, y1, y2;
  int flag=0;
void draw-pixel(int x, int y)
{
   glColor3f(1,0,0);
   glBegin(GL_POINTS);
   glVertex2i(x,y);
     glEnd();
     glFlush();
}
void draw-line()
 {
  int dx, dy, i, e;
  int incx, incy, inc1, inc2;
     int x, y;
   dx = x2-x1;
   dy = y2-y1;
```

Teacher's Signature:

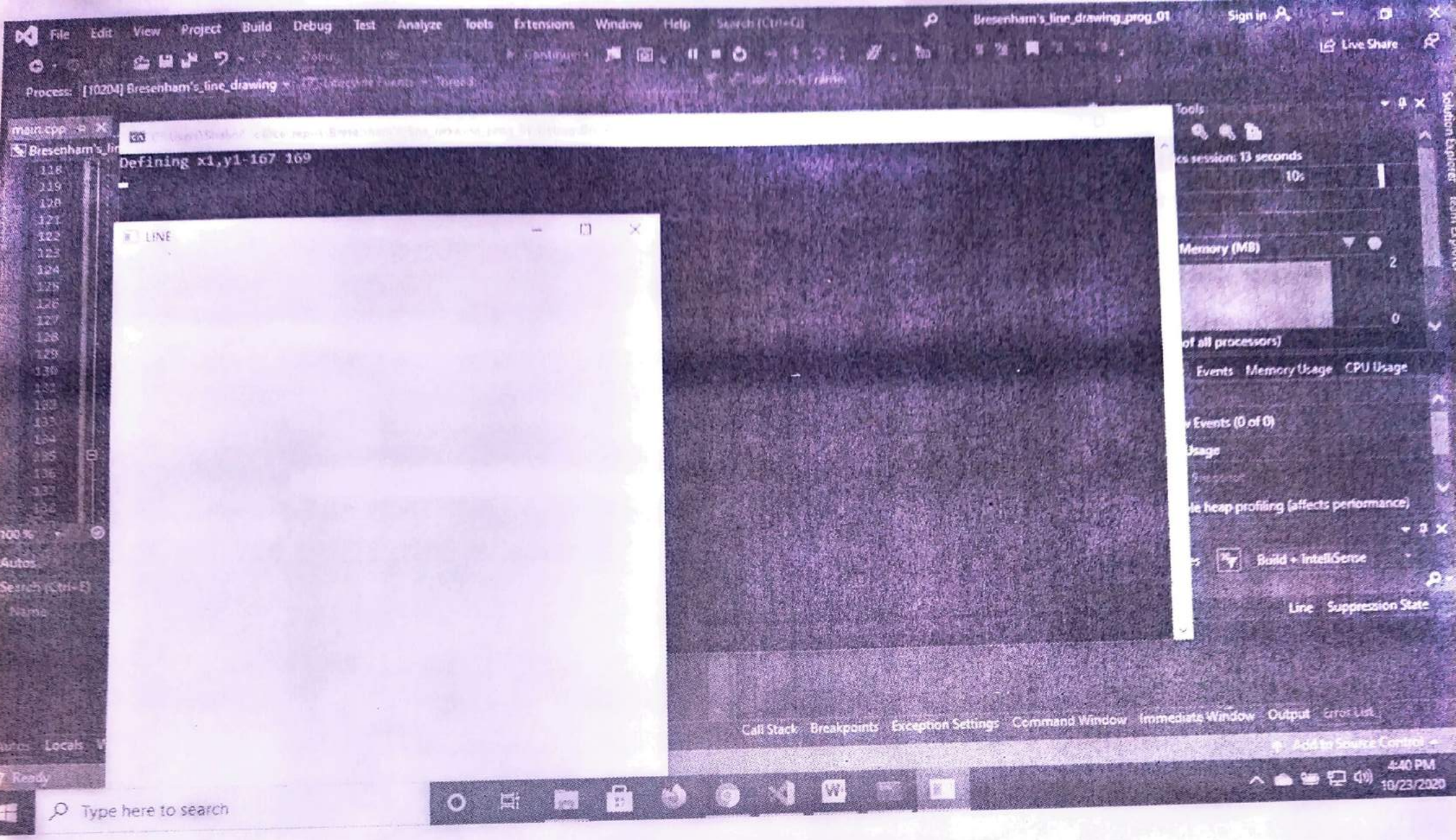
```
if (dx<0) dx = -dx;
if (dy<0) dy = -dy;
    incx = 1;
    if (x2<x1)
        incx = -1;
    incy = 1;
    if (y2<yc1)
        incy = -1;
    x = x1;
    y = yc1;
    if (dx>dy)
{
    draw_pixel(x, y);
     e = 2*dy-dx;
     inc1 = 2*(dy-dx);
     inc2 = 2*dy;
    for(i=0; i<dx; i++)
{
     if (e>0)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

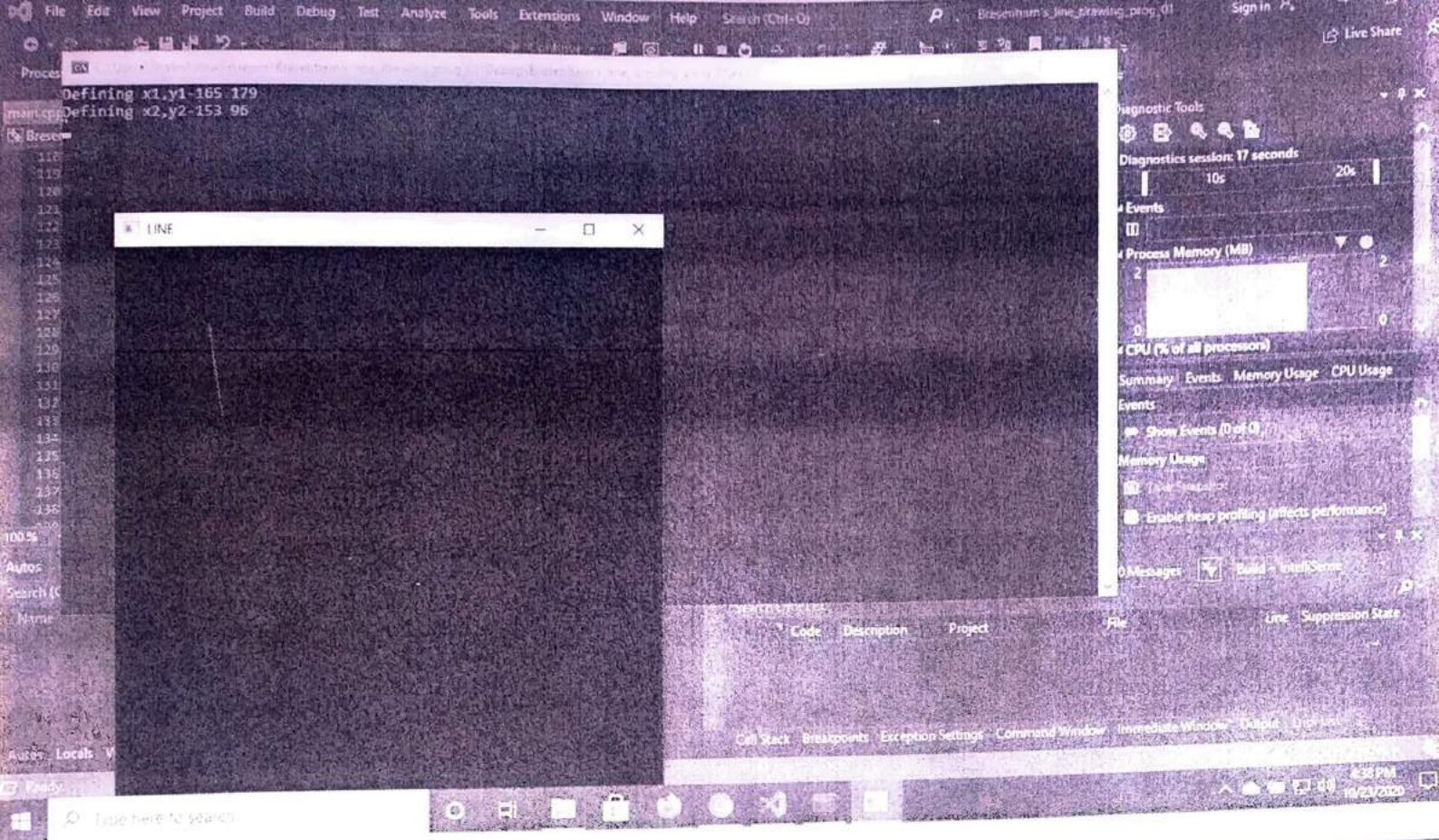
```
else {
    draw_pixel(x,y);
     e = 2*dx-dy;
     incl = 2*(dx-dy);
     inc2 = 2*dx;
     for(i=0; i<dy; i++)
{
  if (e>0)
    {
       x+= incx;
       e+ = incl;
    } else
          e+=inc2;
       y+=incy;
      draw_pixel(x,y);
} }
  glFlush(); }
void myinit()
{
  glClear(GL_COLOR_BUFFER_BIT);
   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;
            yc1 = 250 - y;
            flag ++;
        @   cout<<x1<<" "<<yc1<<"\n";
        }
        else
        {
            printf(" Defining x2, y2");
            x2 = x-250;
            y2 = 250-y;
            flag = 0;
            cout <<x2<< " " <<y2<< "\n";
            draw_line();
        } }
        break;
    } }
void display()
{ }
int main(int ac, char* av[])
{
    cout<<" x1\n";
```

Teacher's Signature:

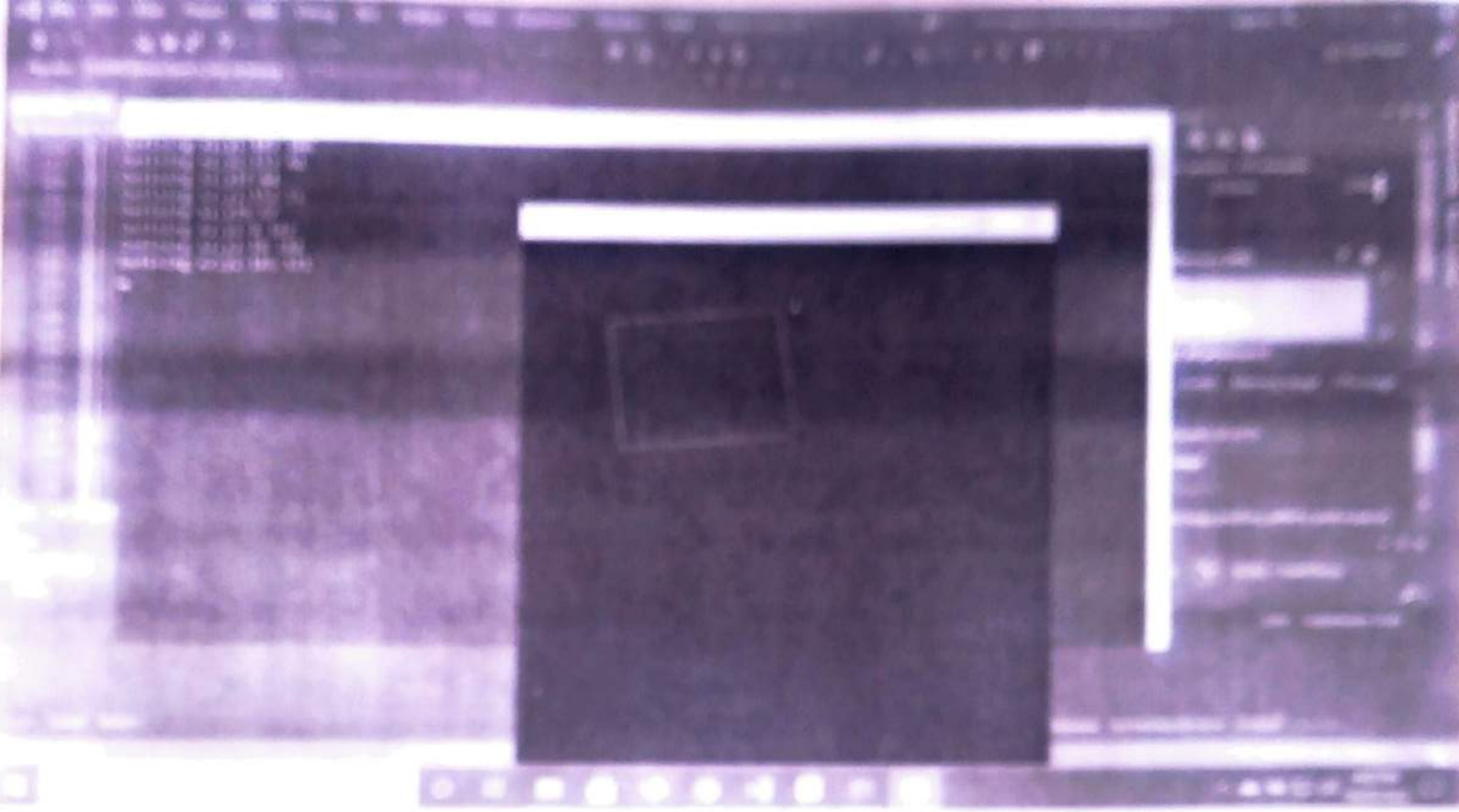
```
cin>>x1;
cout<<"y1\n";
cin>>yc1;
cout<<"x2\n";
cin>>x2;
cout<<"y2\n";
cin>>y2;
glutInit(&ac,av);
glutInitDisplaymode(GLUT_SINGLE|GLUT-RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,200);
glutCreateWindow("LINE");
myinit();
glutMouseFunc(mymouse);
glutDisplayFunc(display);
glutmainloop();
}.
```

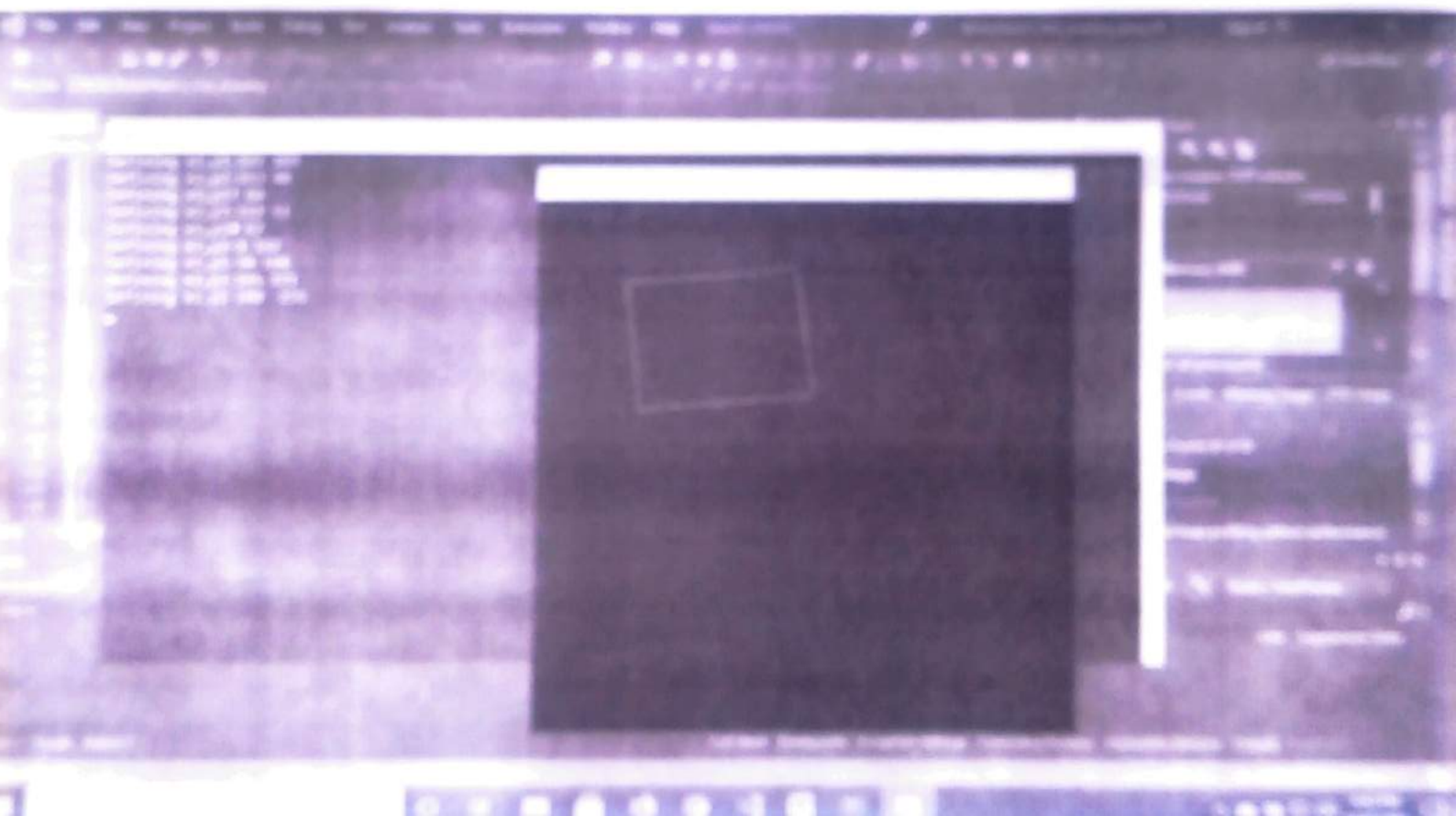

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Bresenham's_line_drawing_prog_01
Sign in
Live Share
Process: [10204] Bresenham's_line_drawing
main.cpp
Defining x1,y1-167 169
LINE
Tools
10s
Memory (MB)
Events Memory Usage CPU Usage
Events (0 of 0)
Build + IntelliSense
Line Suppression State
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output
Ready
Type here to search
4:40 PM
10/23/2020

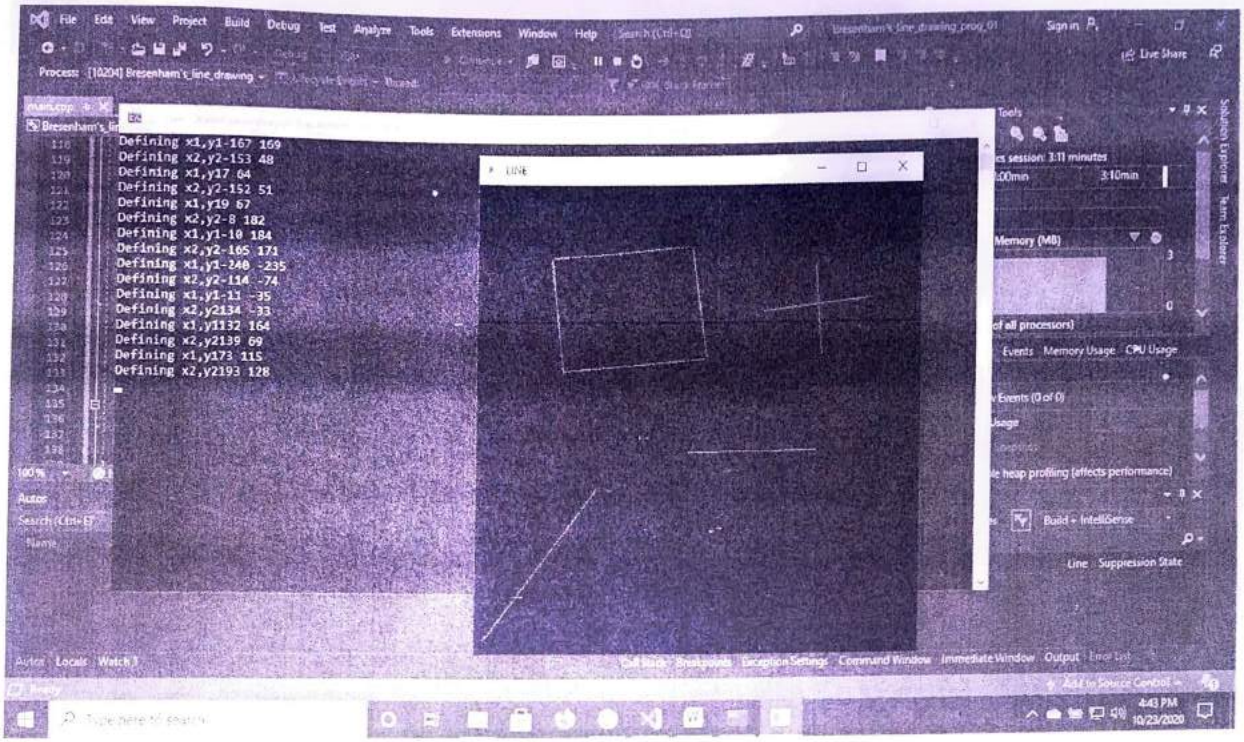

Defining x1,y1-165 179
Defining x2,y2-153 96
LINE


Defining x1,y136 107
Defining x2,y2-113 -63
LINE


Defining x1,y1-167 169
Defining x2,y2-153 48
Defining x1,y17 64
Defining x2,y2-152 51
LINE
Defining x1,y1-167 169
Defining x2,y2-153 48
Defining x1,y17 64
Defining x2,y2-152 51
Defining x1,y19 67
Defining x2,y2-8 182
LINE






Defining x1,y1-167 169
Defining x2,y2-153 48
Defining x1,y17 64
Defining x2,y2-152 51
Defining x1,y19 67
Defining x2,y2-8 182
Defining x1,y1-10 184
Defining x2,y2-165 171
Defining x1,y1-240 -235
Defining x2,y2-114 -74
Defining x1,y1-11 -35
Defining x2,y2134 -33
Defining x1,y1132 164
Defining x2,y2139 69
Defining x1,y173 115
Defining x2,y2193 128
LINE

Experiment Name / No.: 2 - Circle & ellipse drawing

Camlin Page No. 6
Date / /

(2) Write a program to generate a circle & ellipse using Bresenham's circle drawing & ellipse drawing techniques use two windows to draw circle in one window & 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;
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;
```

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 mymouseFunccircle (int button, int state, int x, int y)
{
if (button==GLUT_LEFT_BUTTON&& STATE==GLUT_DOWN&&
                point1_done==0)
  {  P1_x = x-250;
     P1_y = 250-y;
     point1_done = 1;
  }
else if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
  { P2_x = x-250;
    P2_y = 250-y;
     xc = P1_x;
     yc = P1_y;
  float exp=(P2_x-P1_x)*(P2_x-P1_y)*(P2_y-P1_y);
     circlebres();
     point1_done = 0;
  }
}
```

Teacher's Signature:

```
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 midptellipse()
{
  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)
{
  draw_ellipse(xce, yce, x, y);
  if(d1<0)
  {
    x++;
    dx = dx + (2*ry*ry);
    d1 = d1+dx+(ry*ry);
  } else
  {
    x++;
    y--;
```

Teacher's Signature:

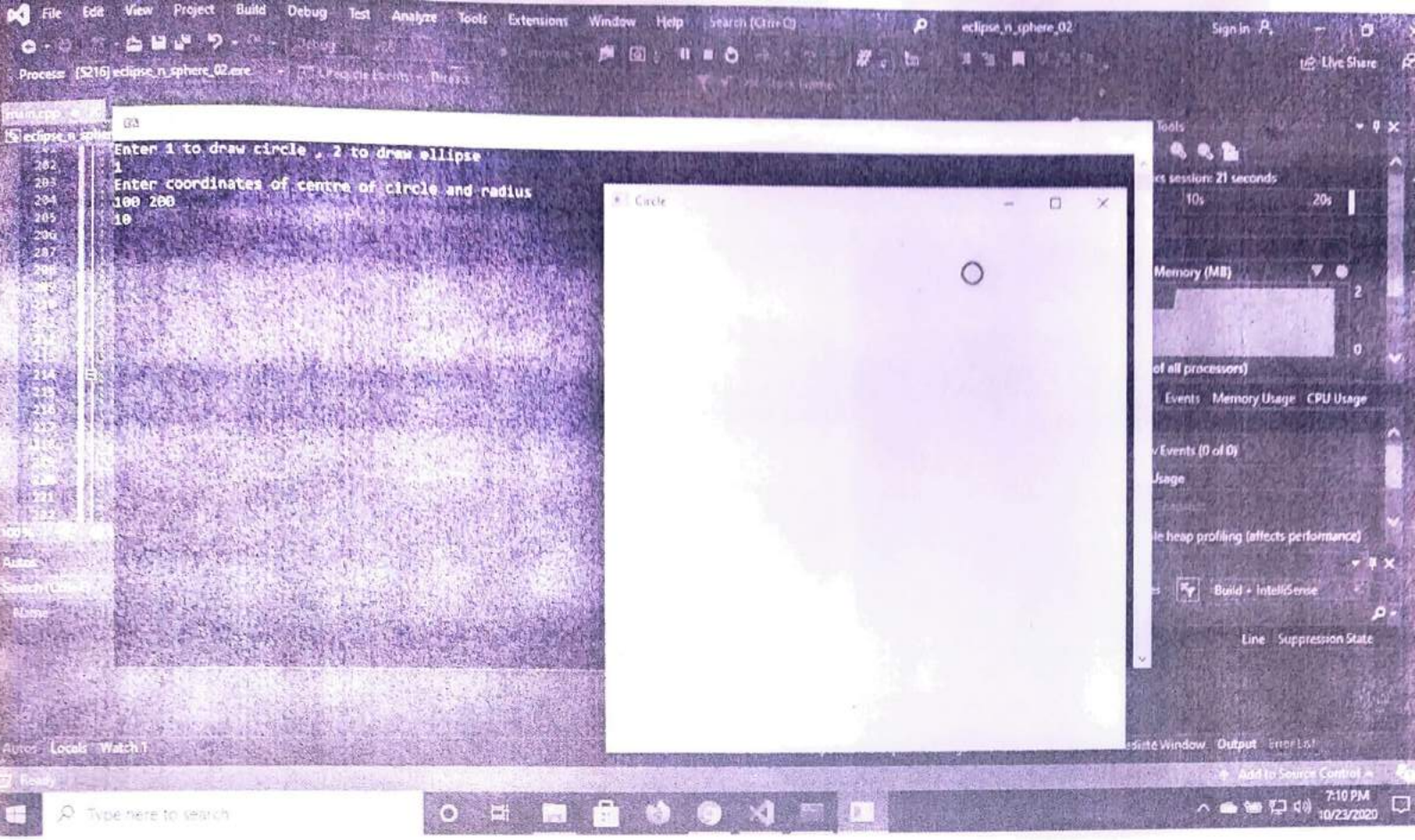
```
dx = dx + (2*ry*ry);
dy = dy - (2*rx*rx);
d1 = d1 + dx - dy + (ry*ry);
}}
while (y>=0)
{ if (d2>0)
  { y--;
    dy = dy - (2*rx*rx);
    d2 = d2 + (rx*rx) - dy;
  } else {
      y--;
      x++;
    dx = dx + (2*ry*ry);
    dy = dy - (2*rx*rx);
     d2 = d2 + dx - dy + (rx*rx);
  }}
  glFlush(); }
int P1e_x, P2e_x, P1e_y, P2e_y, P3e_x, P3e_y;
  int pointle_done = 0;
void myMouseFunc(int button, int state, int x, int y)
{
 if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN &&
       pointle_done == 0) {
     P1e_x = x - 250;
     P1e_y = 250 - y;
       xce = P1e_x;
       yce = P1e_y;
 }     Pointle_done = 1;
```

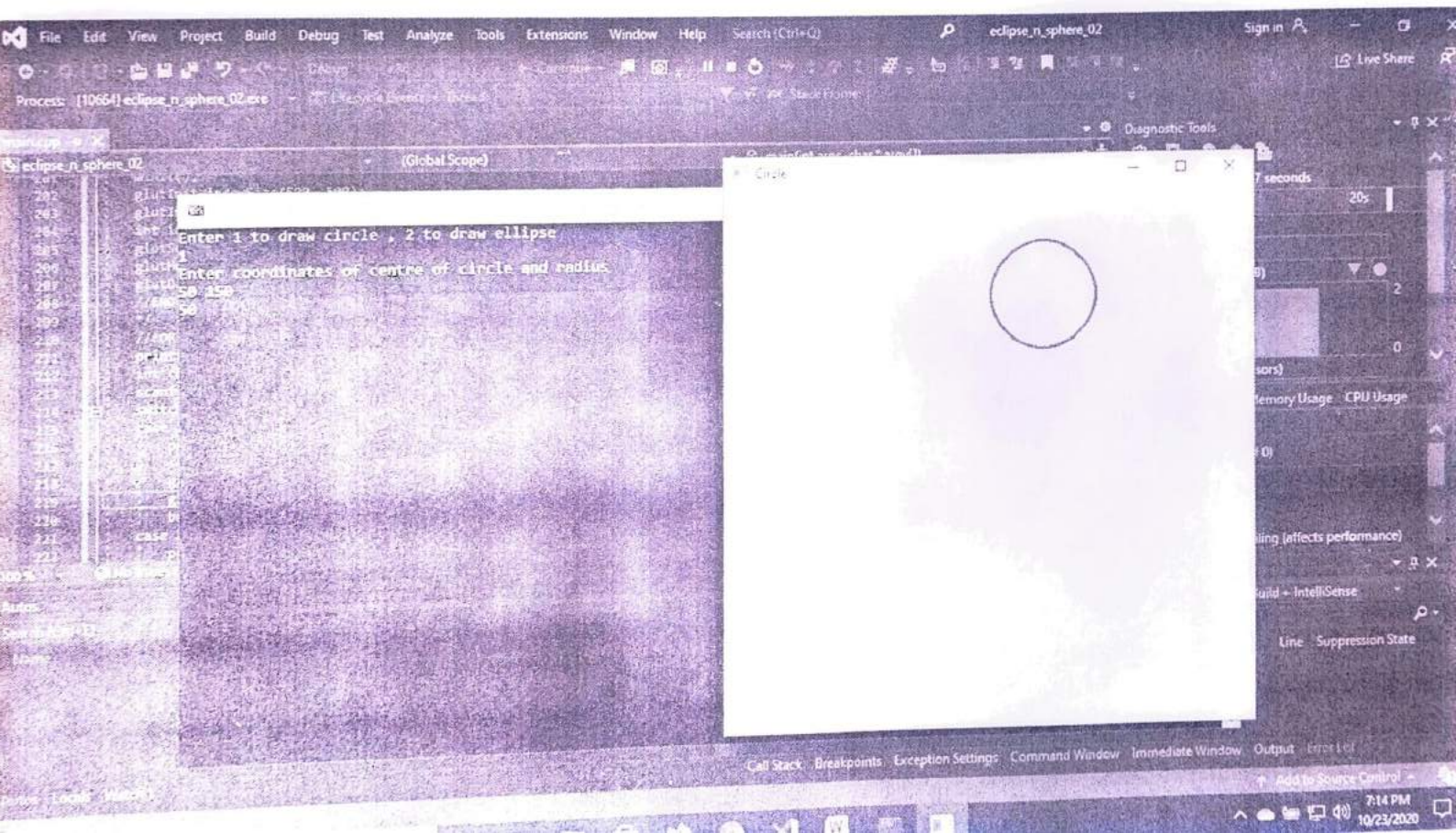
Teacher's Signature:

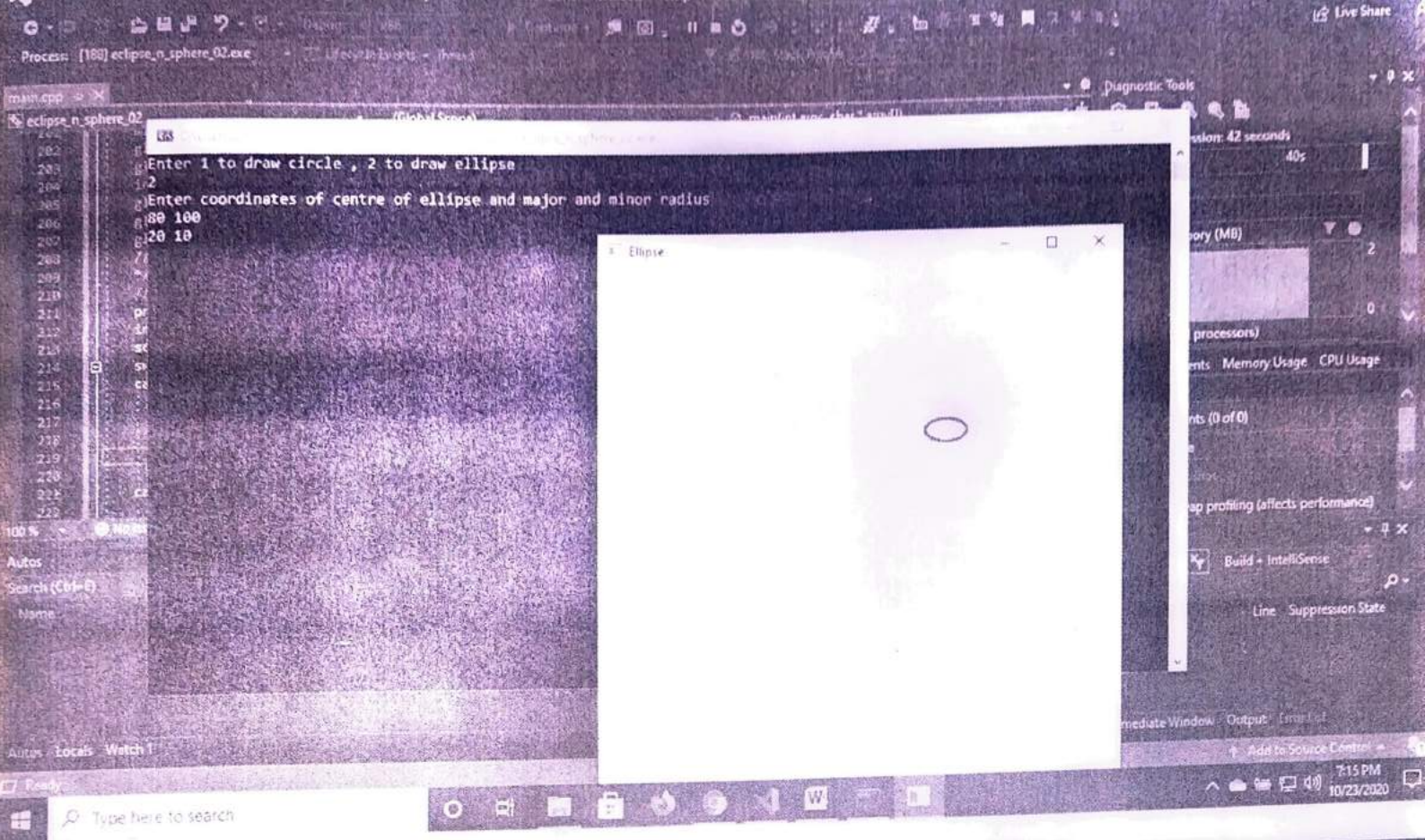
```
else if (button==GLUT_LEFT_BUTTON && state==GLUT_DOWN && pointle_done==2)
{ P3e_x = x-250;
  P3e_y = 250-y;
  float exp=(P3e_x-p1e_x)*(P3e_x-p1e_x)+(P3e_y-p1e_y)*(P3e_y-p1e_y);
  ry=(int)(sqrt(exp));
    midptellipse();
    pointle_done=0;
}

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);
  glutInitWindowPosition(0,0);
 int id1 = glutCreateWindow("Circle");
  glutSetWindow(id1);
  glutMouseFunc(myMouseFunccircle);
  glutDisplayFunc(myDrawing);
    myinit();
 glutInitWindowSize(500,500);
 glutInitWindowsPosition(600,100);
```

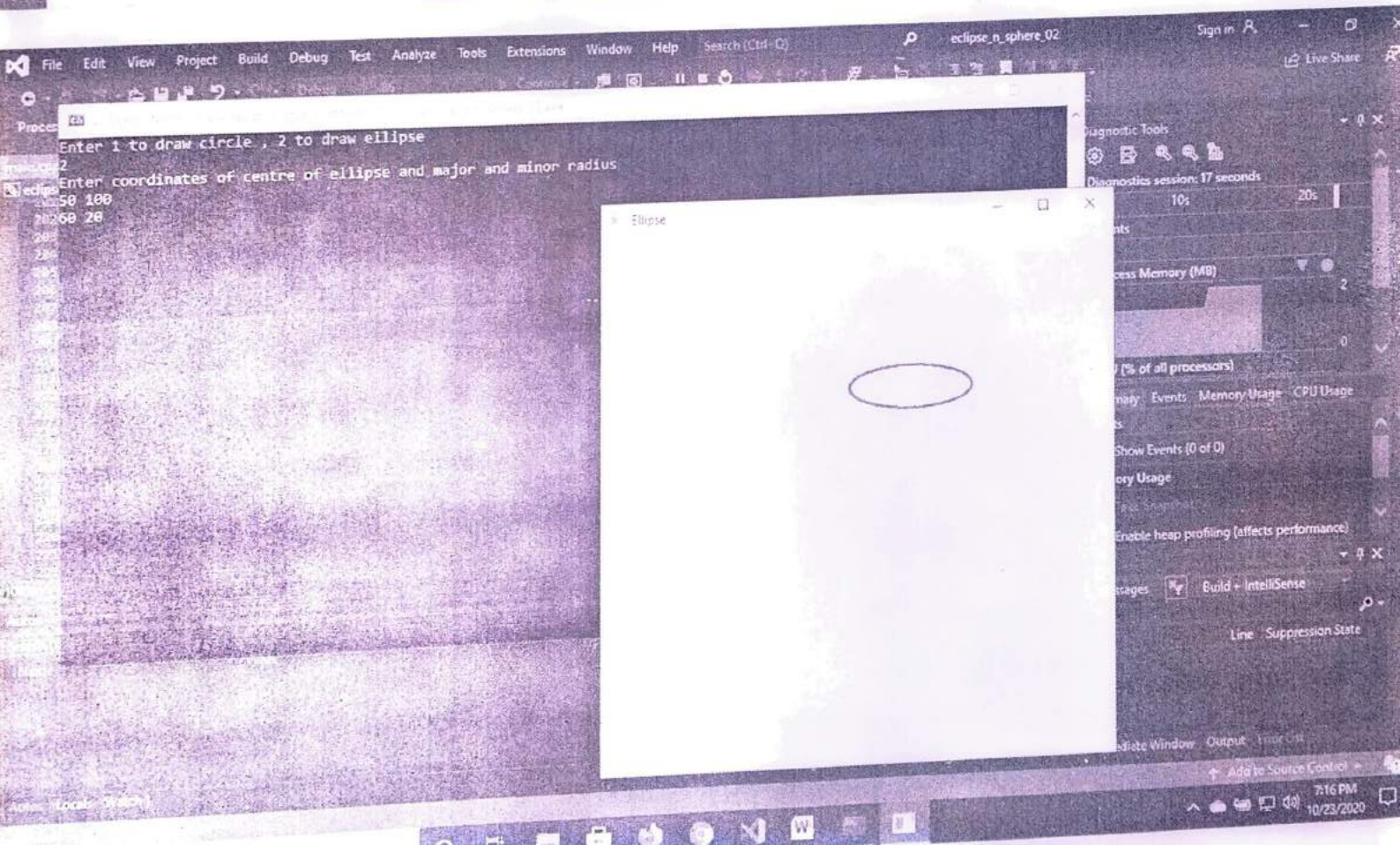
Teacher's Signature:

```
int id2 = glutCreateWindow ("Ellipse");
glutSetWindow (id2);
glutMouseFunc (myMouseFunc);
glutDisplayFunc (myDrawing);
Printf ("Enter 1 to draw circle, 2 to draw ellipse\n");
int ch;
scanf ("%d", &ch);
switch (ch) {
case 1 :
Printf ("Enter co-ordinates of centre of circle & radius\n");
scanf ("%d %d %d", &xc, &yc, &r);
glutCreateWindow ("Circle");
glutDisplayFunc (circlebres);
break;
case 2 :
Printf ("Enter coordinates of centre of ellipse & major\n"
glutCreateWindow ("Ellipse");
glutDisplayFunc (midptellipse);
break;
}
minit ();
glutMainLoop ();
}
```


Enter 1 to draw circle , 2 to draw ellipse
1
Enter coordinates of centre of circle and radius
100 200
10
Circle


Enter 1 to draw circle , 2 to draw ellipse
1
Enter coordinates of centre of circle and radius
Circle


Enter 1 to draw circle , 2 to draw ellipse
2
Enter coordinates of centre of ellipse and major and minor radius
80 100
20 10
Ellipse


Enter 1 to draw circle , 2 to draw ellipse
2
Enter coordinates of centre of ellipse and major and minor radius
50 100
60 20
Ellipse

Experiment Name / No.: 3 - Sierpinski Gasket.

Camlin | Page No. 12 | Date / /

3 Write a program to recursively subdivides a tetrahedron to form a 3D Sierpenski Gasket. The number of recursive steps to be specified at execution time.

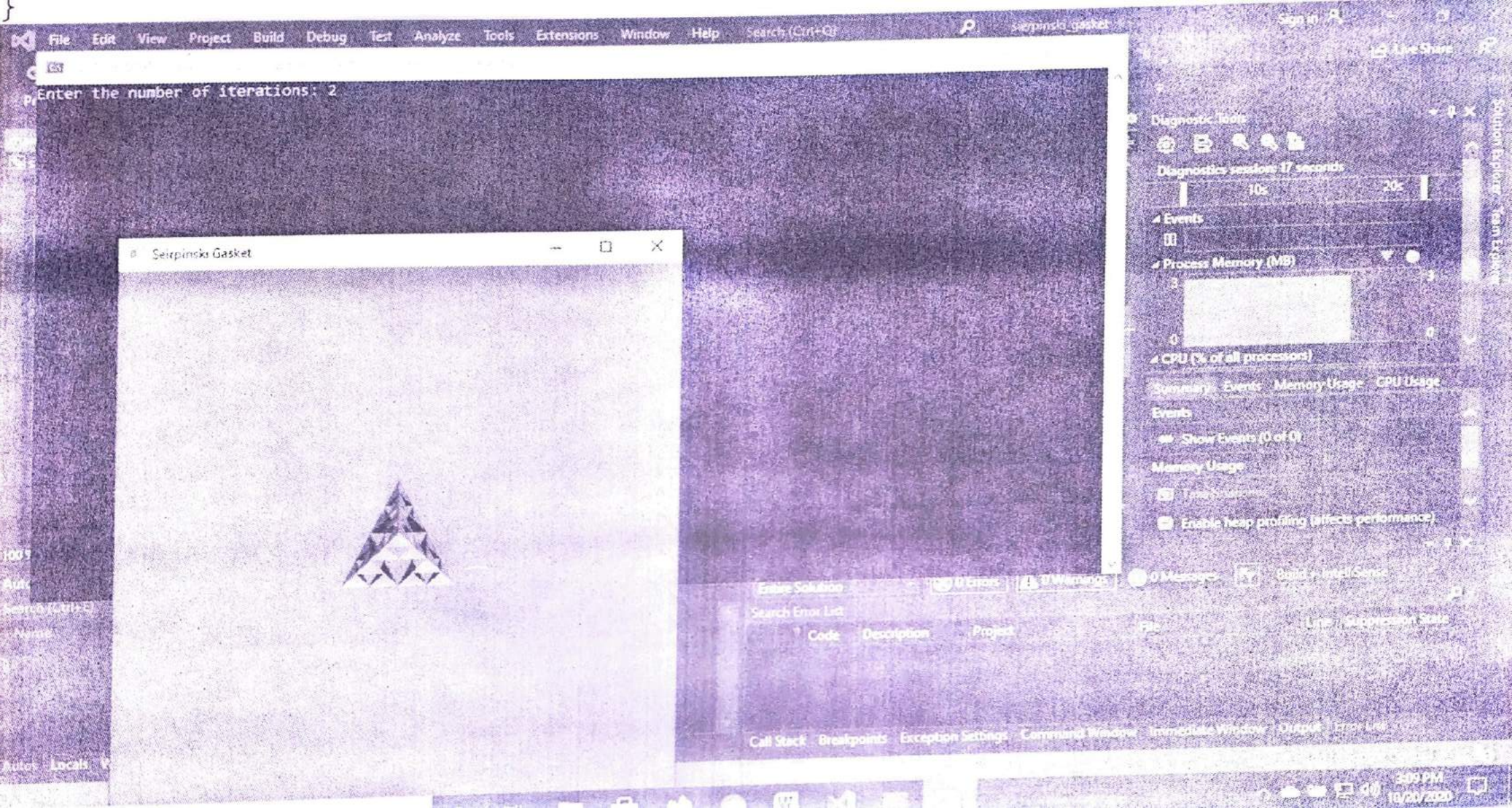
→
```
#include <gl/glut.h>
#include <stdio.h>
int m;
typedef float point[3];
point tetra[4] = { {0,100,-100}, {0,0,100}, {100,-100,-100},
{-100,-100,-100} };
void tetrahedron(void);
void myinit(void);
void divide_triangle(point a, point b, point c, int m);
void draw_triangle(point p1, point p2, point p3);
int main(int argv, char** argc)
{ Printf("Enter the number of iteration:");
  Scanf("%d", &m);
  glutInit(&argv, argc);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
  glutWindowPosition(100,200);
  glutWindowSize(500,500);
  glutCreateWindow("Seirpinski Gasket");
  glutDisplayFunc(tetrahedron);
  glEnable(GL_DEPTH_TEST);
    myinit();
  glutMainLoop();
}
```

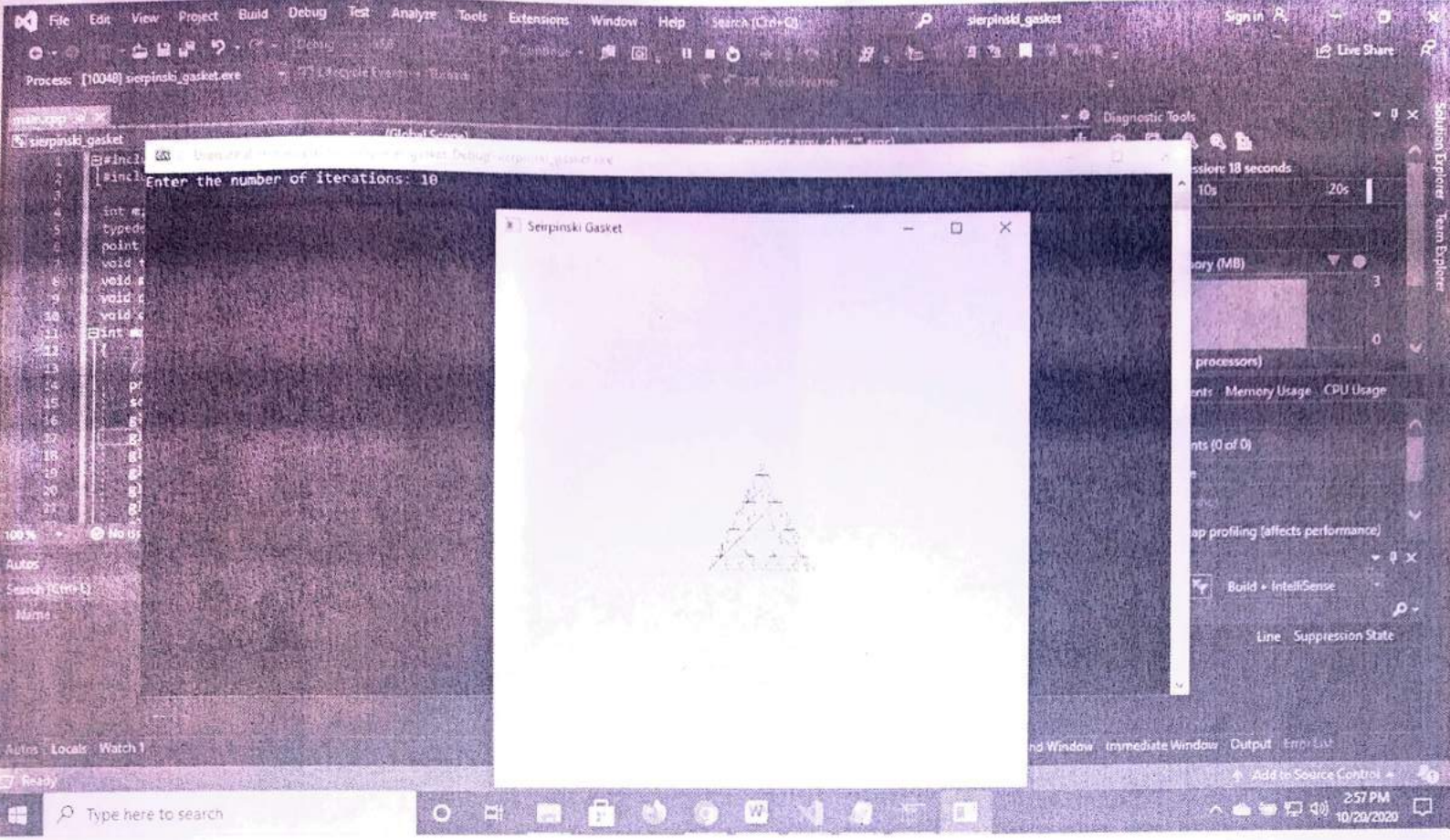
Teacher's Signature:

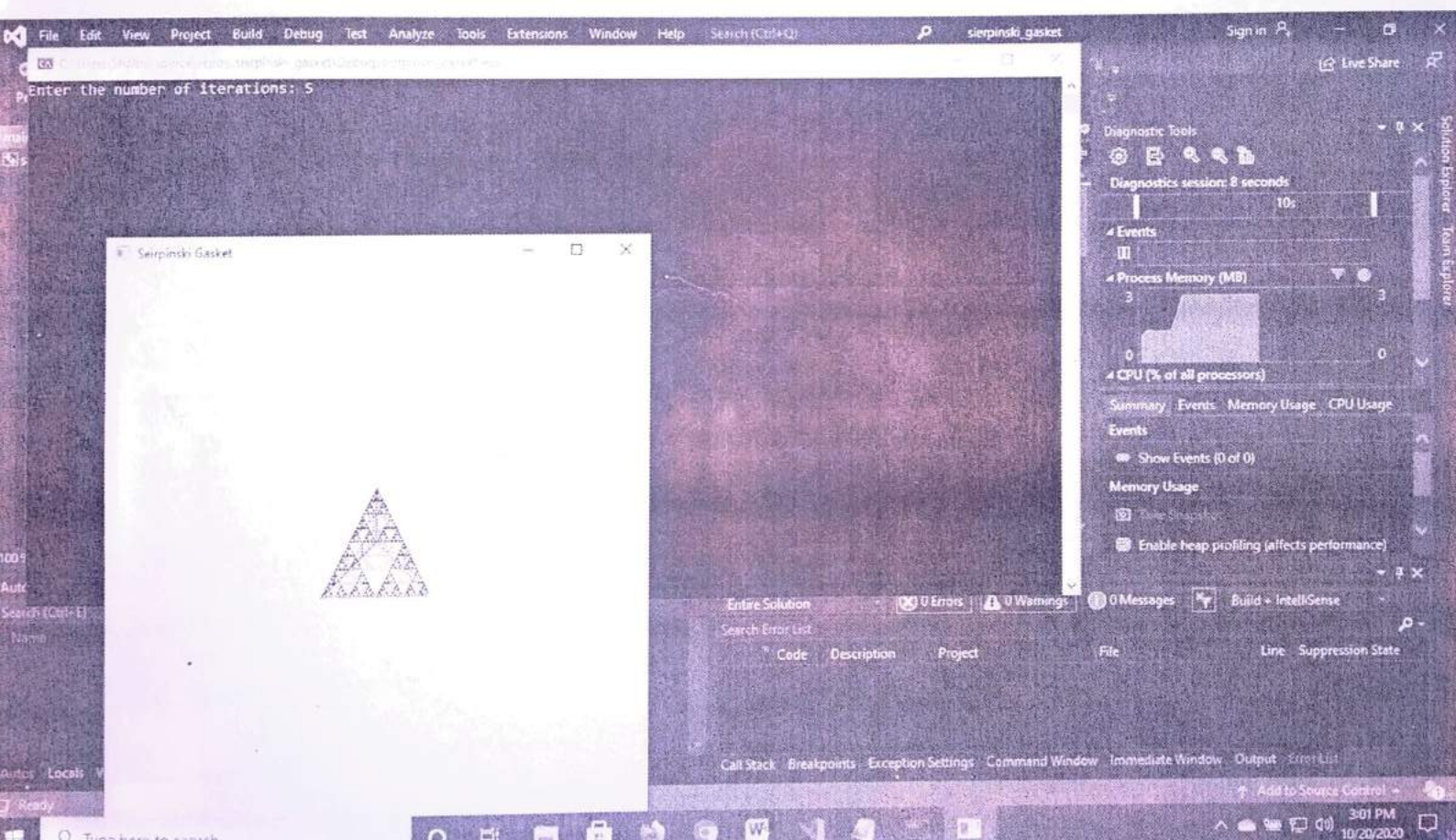
```
void divide_triangle(point a, point b, point c, int m)
{ point v1, v2, v3;
  int j;
  if(m>0) {
    for(j=0; j<3; j++)
      v1[j]=(a[j]+b[j])/2;
    for(j=0; j<3; j++)
      v2[j]=(a[j]+c[j])/2;
    for(j=0; j<3; j++)
      v3[j]=(b[j]+c[j])/2;
    divide_triangle(a, v1, v2, m-1);
    divide_triangle(c, v2, v3, m-1);
    divide_triangle(b, v3, v1, m-1);
  } else
    draw_triangle(a, b, c); }
void myinit() {
    glClearColor(1,1,1,1);
    glOrtho(-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], m);
    glColor3f(0.0, 1.0, 0.0);
  divide_triangle(tetra[3], tetra[2], tetra[1], m);
   glColor3f(0.0, 1.0, 0.0);
  divide_triangle(tetra[0], tetra[2], tetra[3], m);
    glFlush();
}.
```

Teacher's Signature:

```
void draw_triangle (point p1, point p2, point p3)
{
  glBegin (GL_TRIANGLES);
  glvertex3fv (p1);
  glvertex3fv (p2);
  glvertex3fv (p3);
    glEnd ();
}.
```


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Enter the number of iterations: 2
Sierpinski Gasket
Diagnostic Tools
Events
Process Memory (MB)
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Show Events (0 of 0)
Memory Usage
Enable heap profiling (affects performance)
Entire Solution
0 Errors
0 Warnings
Search Error List
Code Description Project File Line
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List
Autos Locals
3:09 PM
10/20/2020


Enter the number of iterations: 10
Sierpinski Gasket


Enter the number of iterations: 5
Sierpinski Gasket

Enter the number of iterations: 2

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];
 y[] = {0,100,50,100,0};
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);
      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;
    }.
```

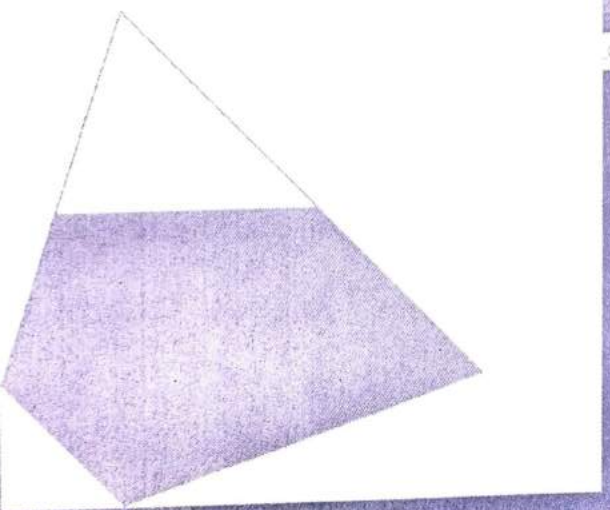
Teacher's Signature:

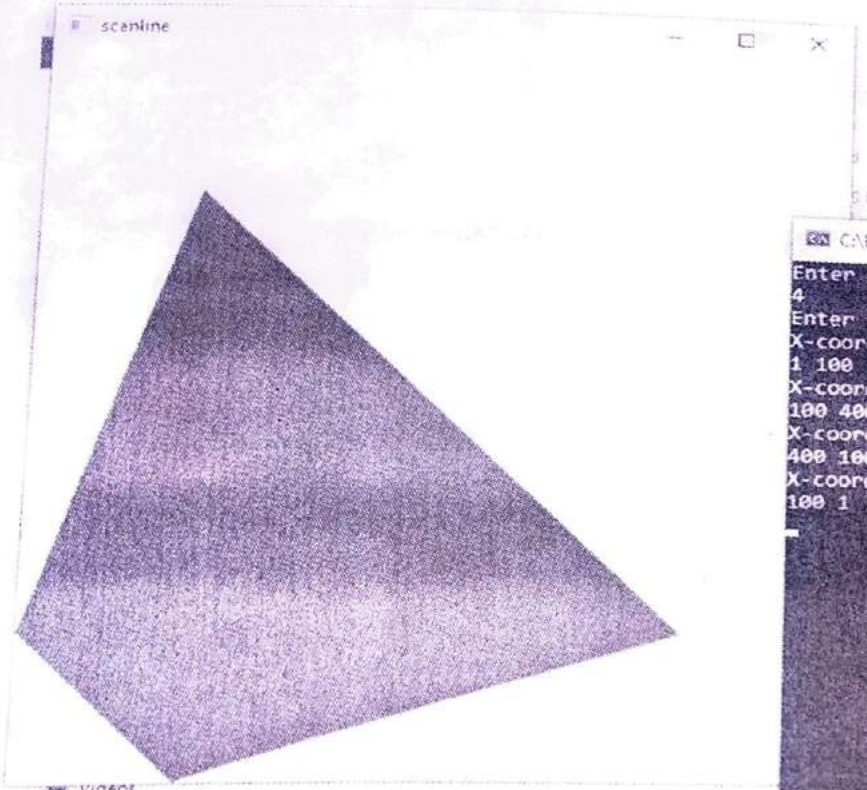
```
if(scanline > y1 && scanline < y2)
        int [m++] = x1 + (scanline - y1) * (x2 - x1) / (y2 - y1); }
void scanfill (float x[], float y[]) {
    for (int s1=0; s1<= wy; s1++) {
                m = 0;
    for (int i=0 ; i<n; i++) {
  edgeDetect(x[i], y[i], x[(i+1)%n], y[(i+1)%n], s1);
}
sort(int x, (int x+m));
        if (m >= 2)
        for (int i=0; i<m; i = i+2)
          draw_line(int x[i], s1, int x[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);
}.
```

Teacher's Signature:

```
void main (int ac, char * av[]) {
    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);
glutInitWindowSize(500, 500);
glutInitWindowPosition (0, 0);
glutCreateWindow("seamline");
glutDisplayFunc (display_filled_polygon);
    myInit();
    glutMainLoop ();
}
```

```
Enter no. of sides:
4
Enter coordinates of endpoints:
X-coord Y-coord:
1 100
X-coord Y-coord:
100 400
X-coord Y-coord:
400 100
X-coord Y-coord:
100 1
```






```
C:\Users\Shalini\source\repos\Scan_Fill_04\Debug\Scan_Fill_04.exe
Enter no. of sides:
4
Enter coordinates of endpoints:
X-coord Y-coord:
1 100
X-coord Y-coord:
100 400
X-coord Y-coord:
400 100
X-coord Y-coord:
100 1
```

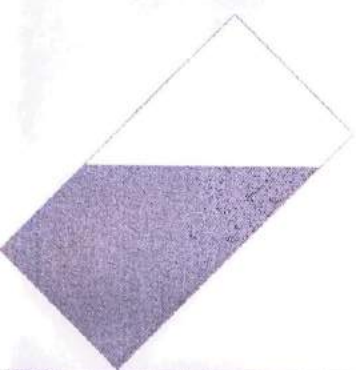
Videos
Local Disk (C:)
Local Disk (D:)
Local Disk (E:)
System Reserved (F:)
New Volume (H:)
Network

16 items 1 item selected 1.85 KB

Type here to search

5:05 PM
10/23/2020

```
Enter no. of sides:
6
Enter coordinates of endpoints:
X-coord Y-coord:
1 100
X-coord Y-coord:
100 200
X-coord Y-coord:
200 300
X-coord Y-coord:
300 200
X-coord Y-coord:
200
100
X-coord Y-coord:
100 1
```



Thank you

Arun Mama

Thanks ma

Vinathi rvce

Yeah... but she said she will take again to watch

scanline

```
Enter no. of sides:
6
Enter coordinates of endpoints:
X-coord Y-coord:
1 100
X-coord Y-coord:
100 200
X-coord Y-coord:
200 300
X-coord Y-coord:
300 200
X-coord Y-coord:
200
100
X-coord Y-coord:
100 1
```

Arun Mama

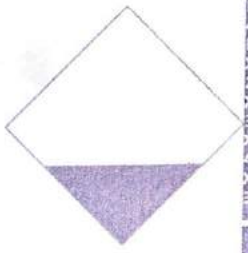
Vinathi rvce

Type here to search

5:18 PM
10/23/2020

scanline

Window Help Search (Ctrl+Q) Scan_Fill_04 Sign in Live Share



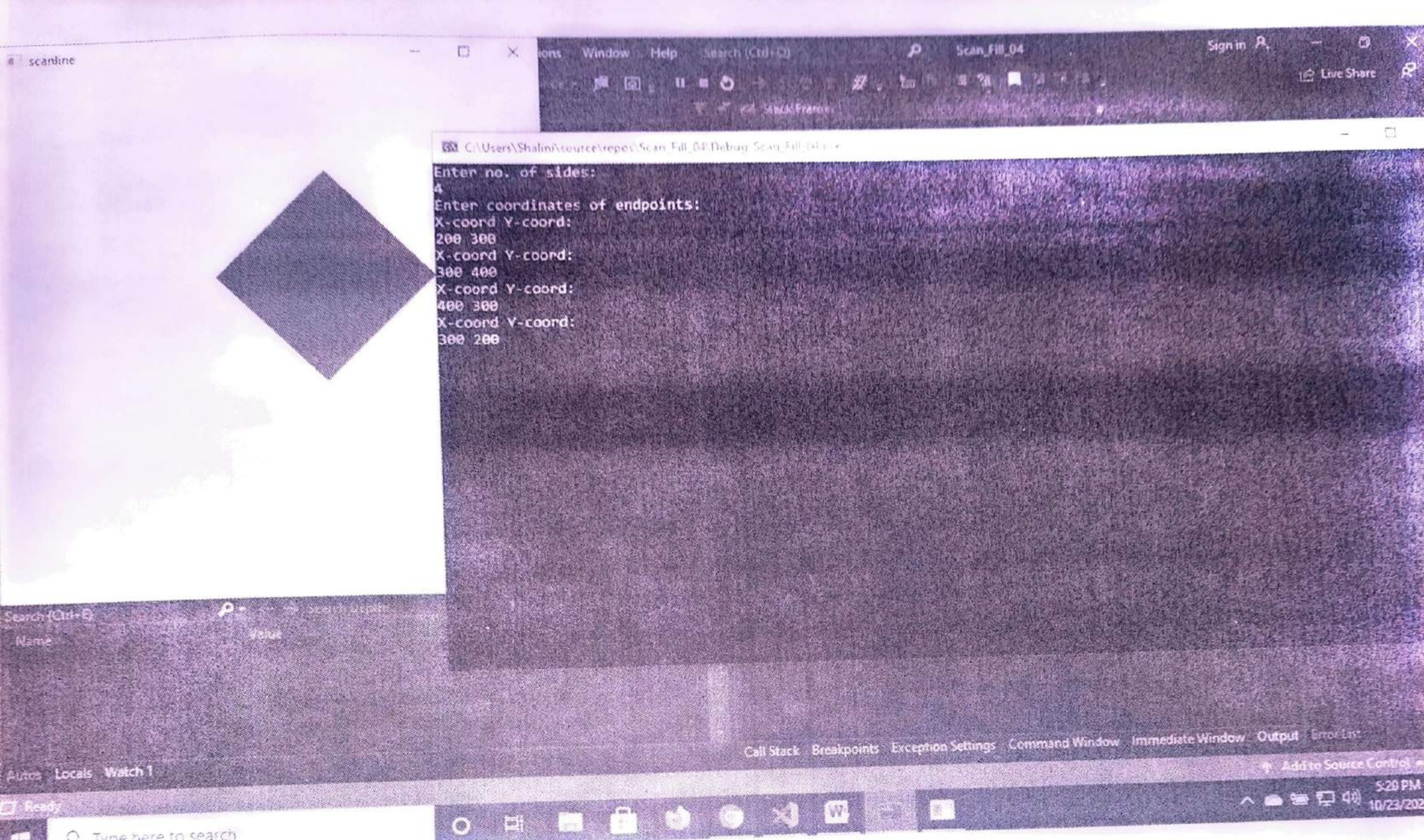
```
Enter no. of sides:
4
Enter coordinates of endpoints:
X-coord Y-coord:
200 300
X-coord Y-coord:
300 400
X-coord Y-coord:
400 300
X-coord Y-coord:
300 200
```

Search (Ctrl+E)

Name

Autos Locals Watch 1

Call Stack Breakpoints Exception Settings


scanline
Scan_Fill_04
Sign in
Live Share
Enter no. of sides:
4
Enter coordinates of endpoints:
X-coord Y-coord:
200 300
X-coord Y-coord:
300 400
X-coord Y-coord:
400 300
X-coord Y-coord:
300 200
Search (Ctrl+E)
Name
Value
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Autos
Locals
Watch 1
Ready
Add to Source Control
5:20 PM

Experiment Name / No.: 5- Rotate, Reflect.

Camlin / Page No. 18.
Date / /

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

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

Teacher's Signature:

```
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();
     glOrtho2D(-450,450,-450,450);
      glMatrixMode(GL_MODELVIEW);
       glLoadIdentity();
       glColor3f(1,0,0);
        glBegin(GL_LINE_LOOP);
       for(int i=0; i<11; i++)
           glVertex2fv(house[i]);
        glEnd();
        glFlush();
  float x1=0, x2=500;
  float y1=m*x1+c;
  float y2=m*x2+c;
```

Teacher's Signature:

```
glColor3f (1,1,0);
glBegin (GL_LINES);
 glVertex2f (x1, Y1);
 glVertex2f (x2, Y2);
    glEnd();
    glFlush();
glPushMatrix();
glTranslatef (0,C,0);
   theta = atan(m);
 theat = theta * 180 / 3.14;
  glRotatef (theta, 0, 0, 1);
  glScalef (1, -1, 1);
  glRotatef (-theta, 0, 0, 1);
   glTranslatef (0, -C, 0);
   glBegin (GL_LINE_LOOP);
 for (int i=0; i < 11; i++)
      glVertex2fv (house[i]);
       glEnd();
       glPopMatrix();
    glFlush(); }
void display2() {
 glClearColor (1,1,1,0);
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode (GL_PROJECTION);
     glLoadIdentity();
    gluOrtho2D (-450, 450, -450, 450);
     glMatrixMode (GL_MODELVIEW);
      glLoadIdentity();
```

Teacher's Signature:

```
glColor3f (1,0,0);
glBegin (GL-LINE-LOOP);
for(int i=0; i<11; i++)
    glVertex2f (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 * 180/3.14;
   glRotatef (theta, 0, 0, 1);
    glScalef (1, -1, 1);
    glRotatef (-theta, 0, 0, 1);
     glTranslatef (0, -C, 0);
    glBegin (GL-LINE-LOOP);
    for(int i=0; i<11; i++)
          glVertex2f(house[i]);
        glEnd();
      glPopMatrix();
      glFlush();
}
```

Teacher's Signature:

```
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 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);
  glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB );
   glutInitWindowSize (400, 400)
   glutInitWindowPosition (100, 100);
    glutCreateWindow ("House Rotation");
     glutDisplayFunc (display);
      glutmouseFunc (mouse);
       myInit ();
        glutmainLoop();
}.
```

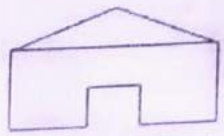
Output:-

C:\Users\hp\source\repos\Lab\Debug\Lab.exe

```
Enter the rotation angle
90
Enter c and m value for line y=mx+c
0
0
```

```
Enter the rotation angle
180
Enter c and m value for line y=mx+c
0
0
```

House Rotation



C:\Users\hp\source\repos\Lab\Debug\Lab.exe

```
Enter the rotation angle
270
Enter c and m value for line y=mx+c
2
3
```

House Rotation

C:\Users\hp\source\repos\Lab\Debug\Lab.exe

```
Enter the rotation angle
360
Enter c and m value for line y=mx+c
2
3
```

6. Write a program to implement the Cohen-Sutherland line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping & viewport for displaying

```
#include<stdio.h>
#include<stdlib.h>
#include<gl/glut.h>
#define outcode int
#define true 1
#define fals 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 Computecode (double x, double y) {
   outcode code = 0;
   if (y > ymax)
         code| = TOP;
   else if (y < ymin)
         code| = BOTTOM;
   if (x > max)
         code| = RIGHT;
}
```

Teacher's Signature:

```
void cohenSutherland(double x0, double y0, double x1, double y1) {
outcode outcode0, outcode1, outcodeout;
bool accept = false, done = false;
outcode0 = computeOutcode(x0, y0);
outcode1 = computeOutcode(x1, y1);
do {
    if (!(outcode0 | outcode1)) {
        accept = true;
        done = true; }
else if (outcode0 & outcode1)
        done = true;
else {
    double x, y;
    outcodeout = outcode0 ? outcode0 : outcode1;
    if (outcodeout & TOP) {
        x = x0 + (x1-x0) * (ymax-y0)/(y1-y0);
        y = ymax; }
    else if (outcodeout & BOTTOM) {
        x = x0 + (x1-x0) * (ymin-y0)/(y1-y0);
        y = ymin; }
    else if (outcodeout & RIGHT) {
        y = y0 + (y1-y0) * (xmax-x0)/(x1-x0);
        x = xmax; }
    else {
        y = y0 + (y1-y0) * (xmin-x0)/(x1-x0);
        x = xmin;
    }
if (outcodeout == outCode0) {
```

Teacher's Signature:

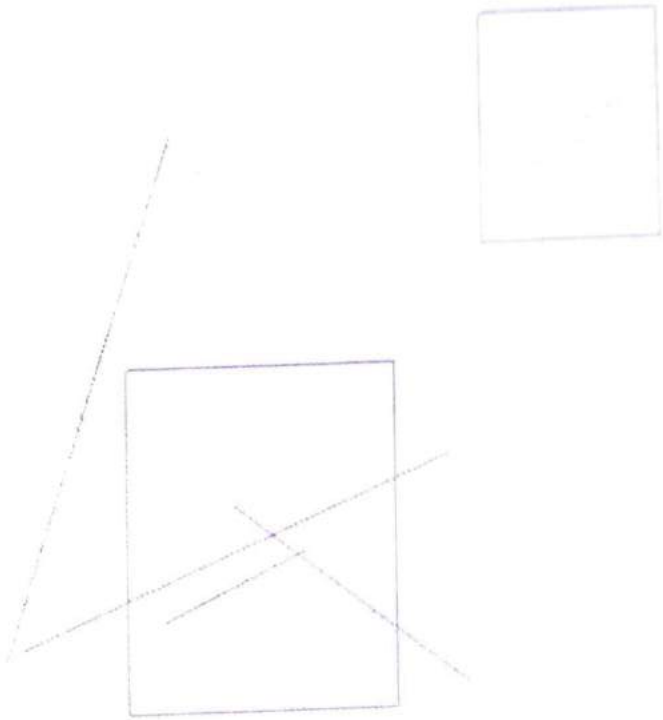
```
x0= x , y0=y , outcode0 = computeoutcode (x0, y0); }
else {
    x1 =x;  y1 =y; outcode1 = compute code (x1,y1); } }
while (! done);
if(accept) {
    double sx= (xvmax - xvmin) / (xmax - xmin);
    double sy = (yvmax - yvmin) / (ymax- ymin);
    double vx0= xvmin + (x0-xmin)* sx;
    double vy0 = 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);
```

Teacher's Signature:

```
void myinit()
{ glClearColor(1,1,1,1);
  glColor3f(1,0,0);
  glPointSize(1.0);
  gluOrtho2D(0,500,0,500); }.
void main(int argc, char **argv) {
printf("Enter window co-ordinates(xmin ymin xmax ymax):\n");
scanf("%lf%lf%lf%lf", &xmin, &ymin, &xmax, &ymax);
printf("Enter viewport coordinates(xvmin yvmin xvmax yvmax)
printf("Enter no of lines:\n");
scanf("%d", &n);
for(int i=0; i<n; i++) {
 printf("Enter line end points(x1,y1 x2 y2):\n");
scanf("%d%d%d%d", &x1[i].x1, &ys[i].y1); }
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(500,500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("clip");
    myinit();
  glutDisplayFunc(display);
      glutMainLoop();
}.
```

Teacher's Signature:

```
Enter window coordinates (xmin ymin xmax ymax):
100 100 250 250
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
300 300 400 400
Enter no. of lines:
4
Enter line endpoints (x1 y1 x2 y2):
120 140 200 170
Enter line endpoints (x1 y1 x2 y2):
40 130 280 210
Enter line endpoints (x1 y1 x2 y2):
80 125 125 350
Enter line endpoints (x1 y1 x2 y2):
150 190 290 110
```



Experiment Name / No.: 7 - Liang-Barsky line clipping algorithm

Camlin Page No. 28
Date / /

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

```
#include <stdio.h>
#include <GL/glut.h>
double xmin, ymin, xmax, ymax;
double wxmin, wymin, wxmax, wymax;
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;
 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);
 outside return(true); }
void liangBarsky(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(xvmax, yvmin);
glVertex2f(xvmax, 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;
```

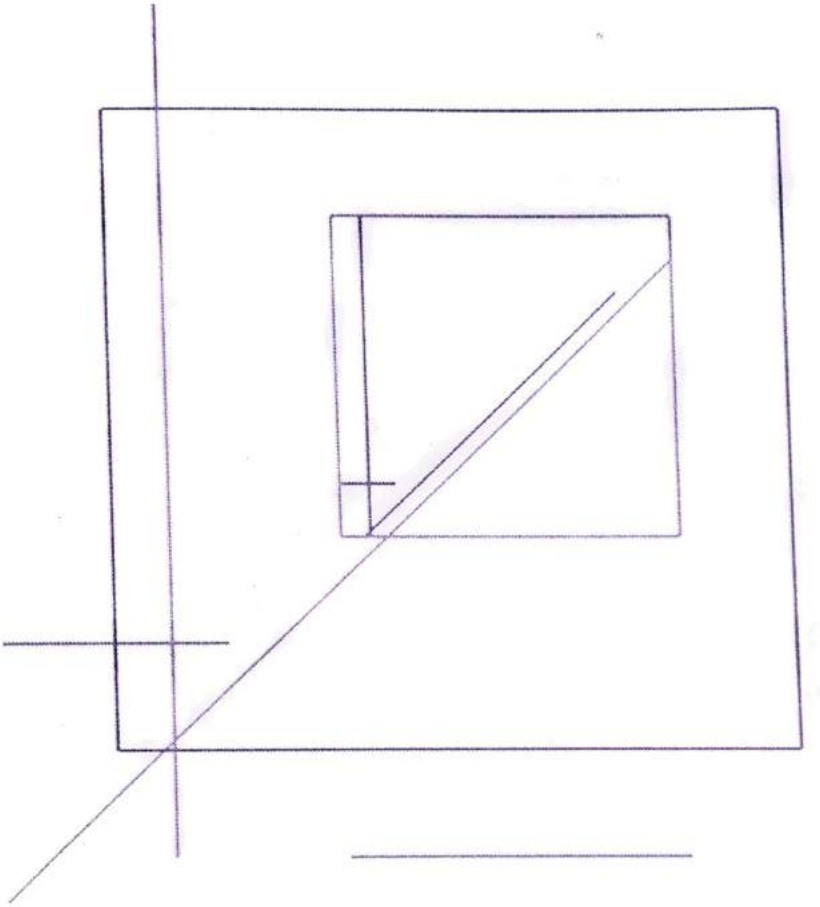
Teacher's Signature:

```
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(); }
void myinit() {
glclearcolor (1.0,1.0,1.0,1.0);
glcolor3f(1.0,0.0,0.0);
glLinewidth (2.0);
glmatrix mode(GL_PROJECTION);
glloadIdentity();
gluOrtho2D (0.0, 499.0,0.0,499.0);}
int main (int argc, char **arg v)
{ glutInit (&argc, arg V);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
Printf("Enter window Coordinates)
Printf("Enter viewport Coordinates);
scanf("%f %f %f %f", &
xvmin, &yvmin, &xvmax, &yvmax);
Printf("Enter the no of lines \n");
scanf("%d", &n);
for(int i=0; i<n; i++) {
Printf("Enter co-ordinates);
scanf("%d %d %d %d", &ls[i].x1,
&ls[i].y1, &ls[i].x2, &ls[i].y2);
}
glutCreatewindow("Liang
Barsky line clipping");
glutdisplayFunc(display);
myinit();
glutmainLoop();
}
```

Teacher's Signature:

```
Enter window coordinates: (xmin ymin xmax ymax)
100 100 400 400
Enter viewport coordinates: (xvmin yvmin xvmax yvmax)
200 200 350 350
Enter no. of lines:
4
Enter coordinates: (x1 y1 x2 y2)
200 50 350 50
Enter coordinates: (x1 y1 x2 y2)
50 30 350 330
Enter coordinates: (x1 y1 x2 y2)
50 150 150 150
Enter coordinates: (x1 y1 x2 y2)
125 50 125 450
```

Liang Barsky Line Clipping Algorithm



Experiment Name / No.: 8 - Cohen - Hodgeman Polygon clipping.

Camlin / Page No. 30 / Date / /

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

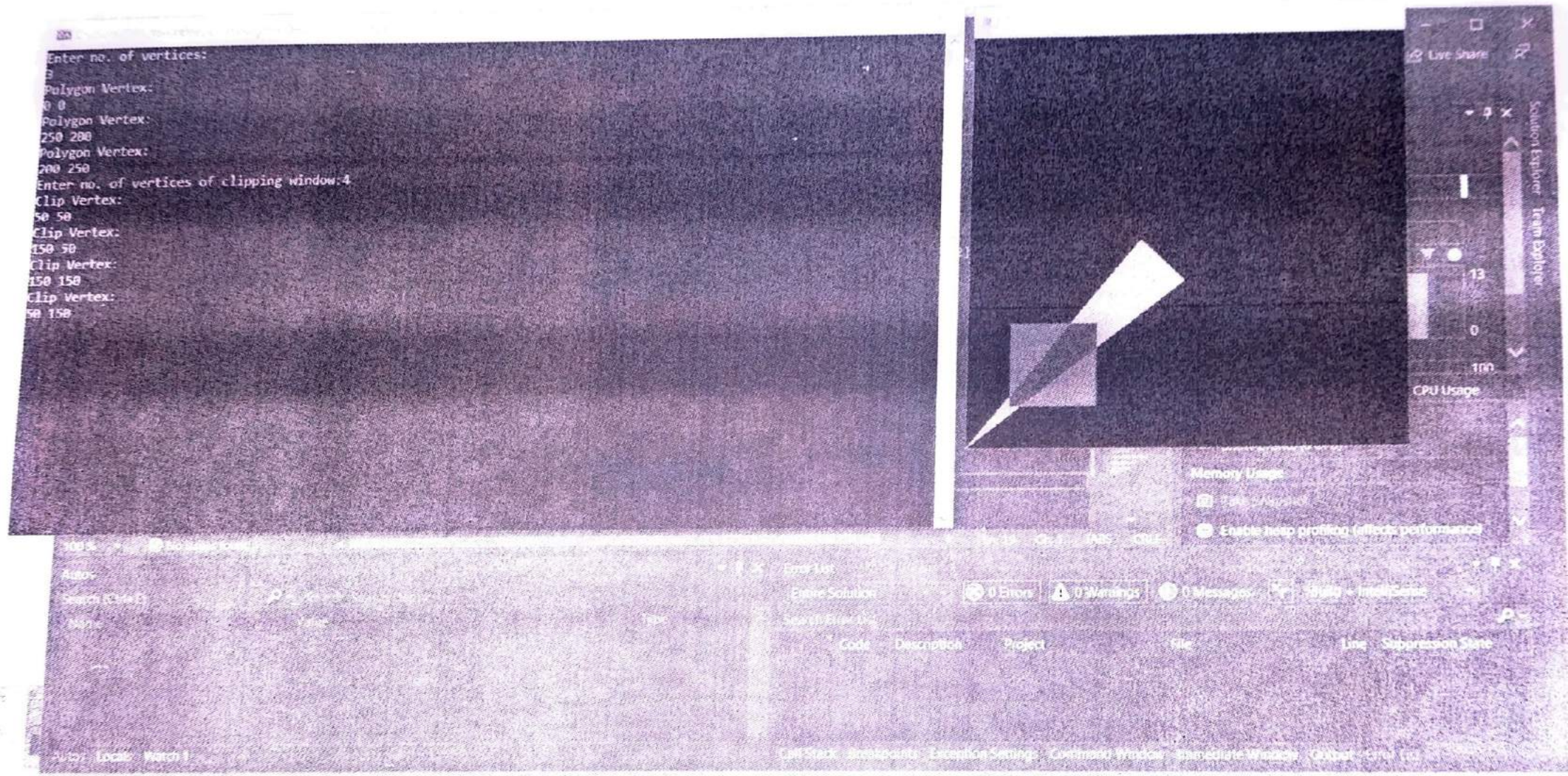
```
#include <iostream>
#include <GL/glut.h>
using namespace std;
int poly_size, poly_points[20][2],
org_poly_size, org_poly_points[20][2]
clipper_size, clipper_points[20][2];
const int MAX_POINTS = 20;
void drawpoly (int P[][2],
int n) {
glBegin(GL_POLYGON);
for(int i=0; i<n; i++)
glVertex2f(P[i][0], P[i][1]);
glEnd(); }
int x_intersect(int x1, int y1,
int x2, int y2, int x3, int y3,
int x4, int y4) {
int num = (x1*y2-y1*x2) *
(x3-x4) - (x1-x2)*(x3*
x4 - y3*x4);
int den = (x1-x2)*(y3-y4) -
(y1-y2)*(x3-x4); }
int y_intersect(int x1, int y1, int
x2, int y2, int x3, int y3, int
x4, int y4) {
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)*(ky-y1) - (y2-y
*(kx-x1);
if(i_pos>=0 && k_pos>=0)
{
new_points[new_poly_size][0] = k
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, k
new_poly_size++;
```

Teacher's Signature:

```
else {
    }
}
poly_size = new_poly_size;
for(int i=0; i<poly_size; i++) {
poly_points[i][0] = 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);
glClear(GL_COLOR_BUFFER_BIT);
  }
void display()
{
  init();
glColor3f(1.0f, 0.0f, 0.0f);
drawPoly(clipper_points, clipperSize);
glColor3f(0.0f, 1.0f, 0.0f);
drawPoly(org_poly_points,
  org_poly_size);
for(int i=0; i<clipper_size; i++)
  {
int k=(i+1)%clipper_size;
Clip(poly_points, poly_size, clipper
points[i][0], clipper_points
[i][1], clipper_points[k][0],
clipper_points[k][1]); }
}
int main(int argc, char argv)
{
printf("Enter no of vertices \n");
scanf("%d", &poly_size);
org_poly_size = poly_size;
for(int i=0; i<poly_size; i++) {
printf("polygon Vertex : \n");
  scanf_s("%d%d", &poly_points
[i][0], &poly_points[i][1]);
org_poly_points[i][0] = poly_
points[i][0];
  }
Printf("Enter no. of vertices of
clipping window: ");
scanf_s("%d", &clipper_size);
for(int i=0; i<clipper_size; i++)
  {
printf("clip vertex : \n");
scanf_s("%d %d", &clipper_points);
  } glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE)
glutInitWindowSize(400, 400);
glutInitWindowPosition(100, 100);
glutCreateWindow("polygon clipping\n");
 glutDisplayFunc(display);
  glutMainLoop();
return 0;
}
```

Teacher's Signature:

OUTPUT:


Enter no. of vertices:
3
Polygon Vertex:
0 0
Polygon Vertex:
250 200
Polygon Vertex:
200 250
Enter no. of vertices of clipping window:4
Clip Vertex:
50 50
Clip Vertex:
150 50
Clip Vertex:
150 150
Clip Vertex:
50 150
Live Share
Solution Explorer
Team Explorer
13
0
100
CPU Usage
Memory Usage
Enable heap profiling (affects performance)
0 Errors
0 Warnings
0 Messages
Code
Description
Project
File
Line
Suppression State

Experiment Name / No: 9: moving car from end to end.

Camlin | Page No. 32
Date / /

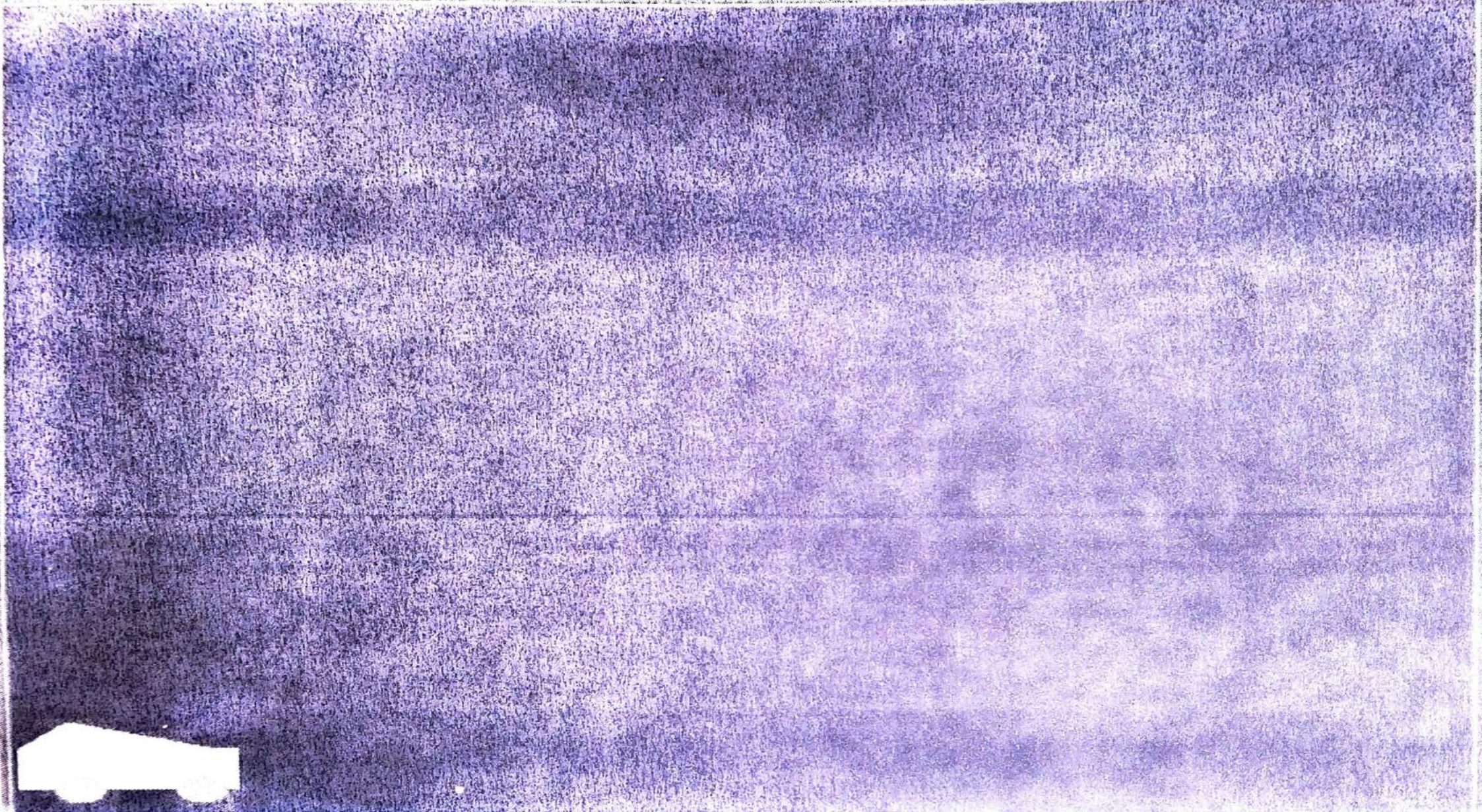
9. Write a program to model a car like using display lists & move a car from one end of the screen to other end. User is able to control the speed with mouse.

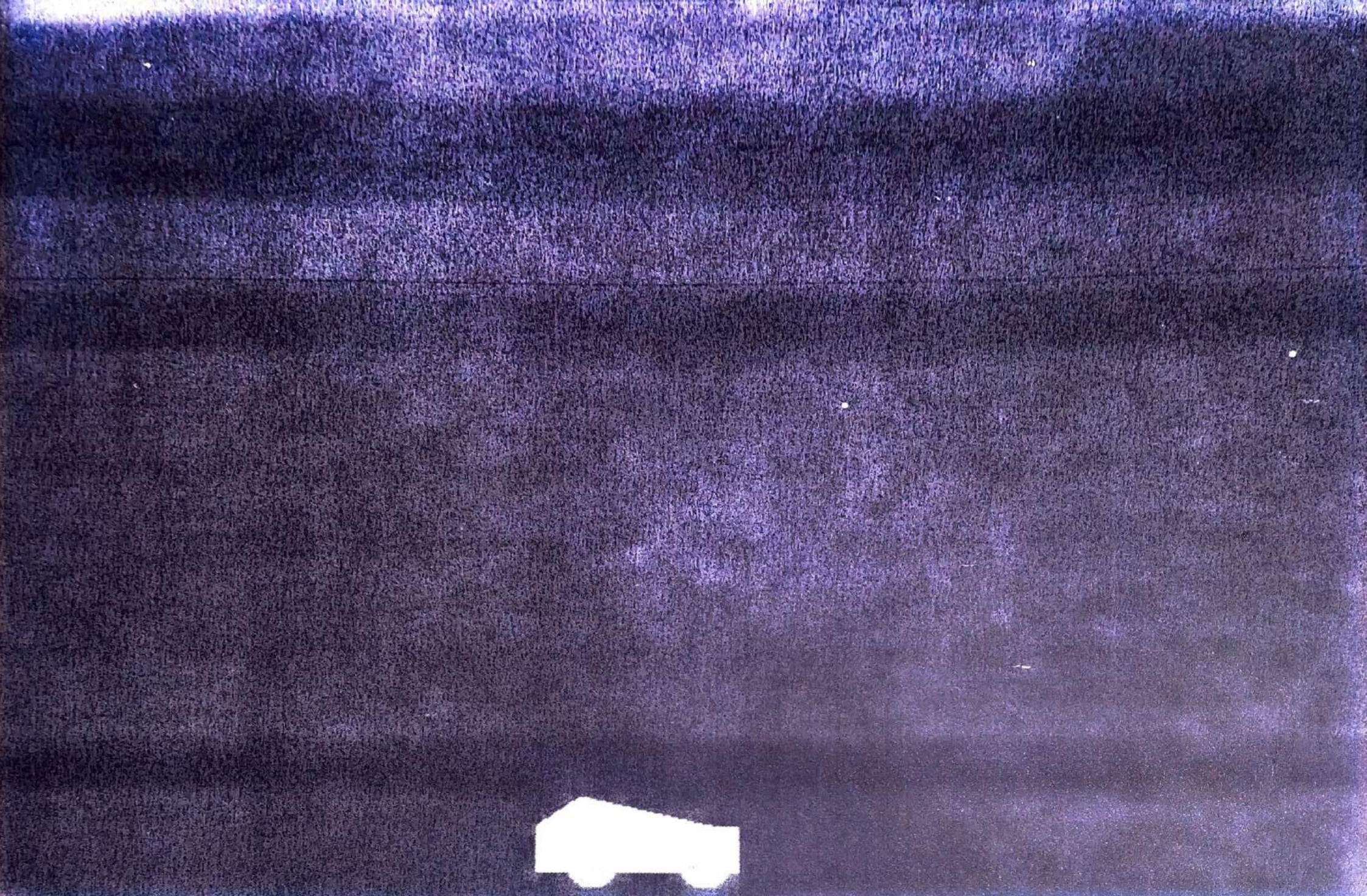
```
#include <GL/glut.h>
#include <math.h>
#includ. <stdio.h>
#define CAR 1
#define WHEEL 2
float S = 1;
void carlist() {
glNewList(CAR, GL_COMPILE);
glColor3f(1,1,1);
glBegin(GL_POLYGON);
glVertex3f(0,25,0);
glVertex3f(90,25,0);
glVertex3f(90,55,0);
glVertex3f(80,55,0);
glVertex3f(20,75,0);
glVertex(0,55,0);
glEnd();
glEndList(); }
Void wheellist() {
glNewList(WHEEL, GL_COM
PILE & EXECUTE);
glColor3f(0,1,1);
glutSolidSphere(10,25,25);
} glEndList();
void wheellist() {
glNewList(WHEEL, GL_COMPILE-
AND_EXECUTE);
glColor3f(0,1,1);
glutSolidSphere(10,25,25);
glEndList();
}
void mykeyboard(unsigned char
key, int x, int y) {
Switch(key) {
case 't': glutPostRedisplay();
break;
case 'q': exit(0);
default: break;
}
}
void myInit() {
glClearColor(0,0,0,0);
glOrtho(0,600,0,600,0,600);
}
void draw_wheel() {
glcolor3f(0,1,1);
glutSolidsphere(10,25,25);
}.
```

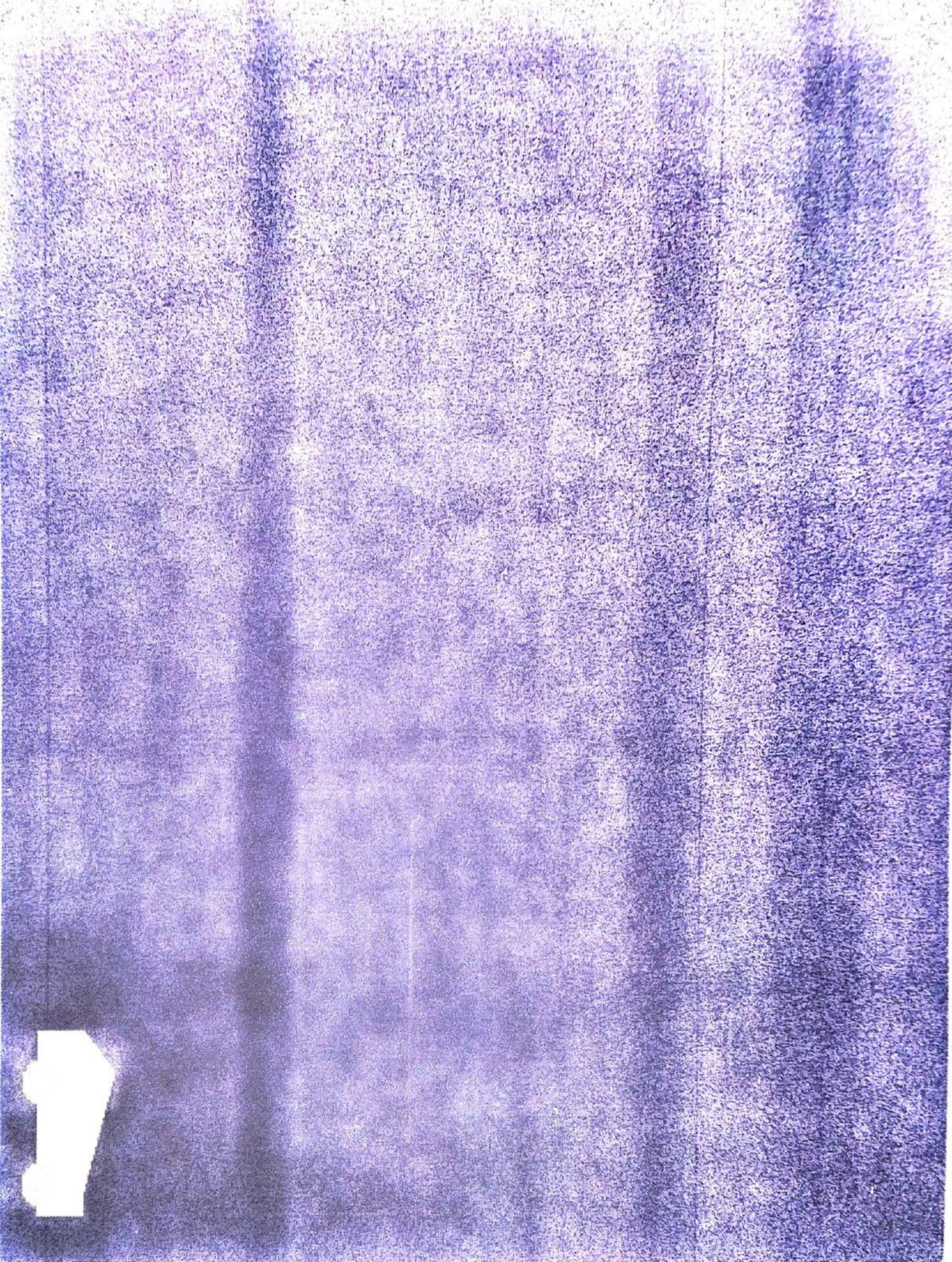
Teacher's Signature:

```
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 main(int argc, char* argV[]) {
     glutInit(&argc, argV);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(600,500);
    glutInitWindowPosition(100,100)
     glutCreateWindow("Car");
      myInit();
      glutDisplayFunc(myDisp);
       glutMouseFunc(Mouse);
        glutKeyboardFunc(myKeyboard);
   }
```

Teacher's Signature:







Experiment Name / No.: 10 - Create a cube & spin.

Camlin Page No. 34
Date / /

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

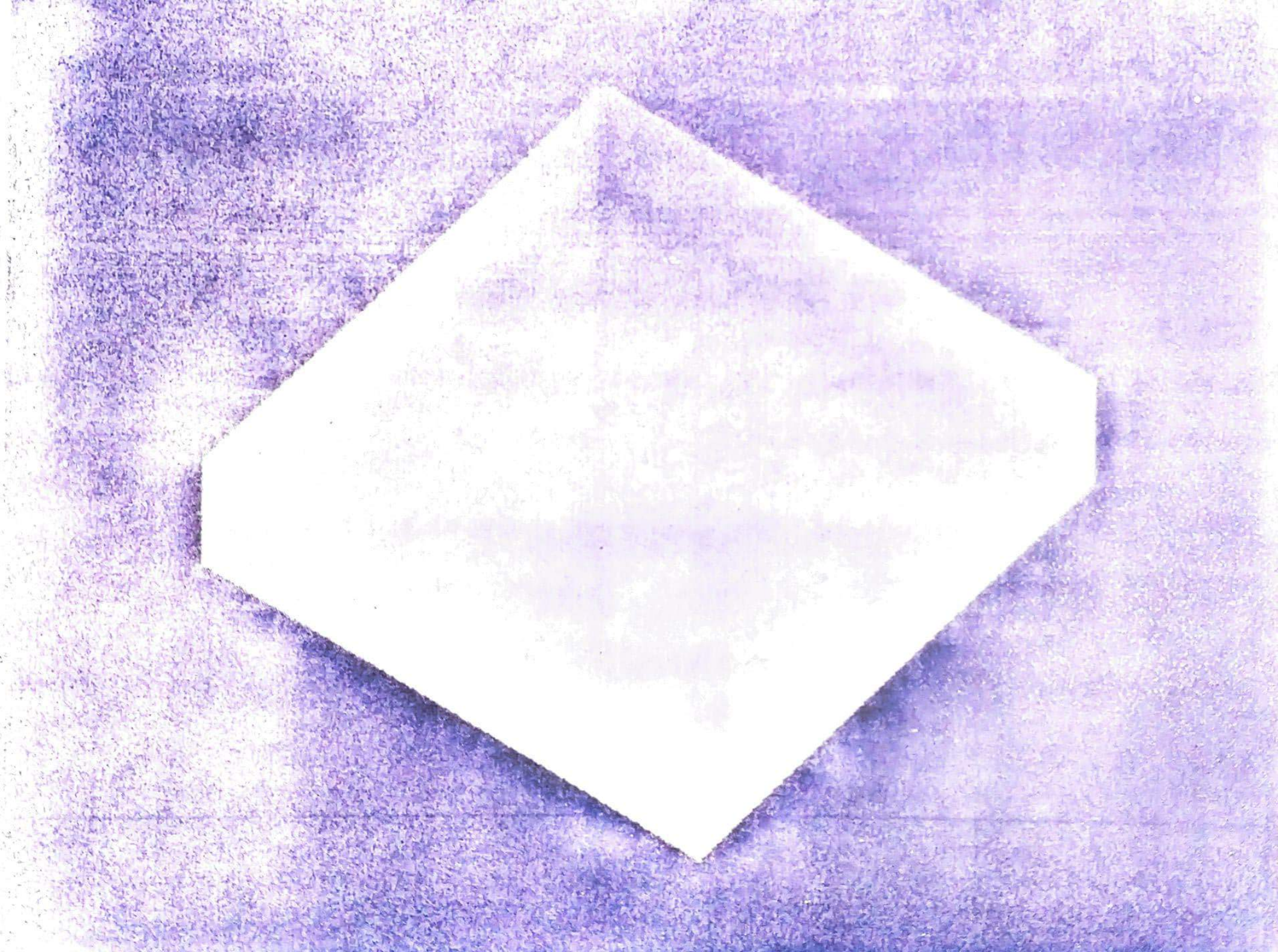
```
#include <stdlib.h>
#include <GL/glut.h>
#include <gl/GL.h>
#include <time.h>
GLfloat vertices[] = {-1.0,-1.0,-1.0, 1.0,-1.0,-1.0, 1.0,1.0,-1.0,
-1.0,1.0,-1.0,-1.0,-1.0,1.0, 1.0,-1.0,1.0, 1.0,1.0,1.0, -1.0,1.0,1.0};
GLfloat colors[] = {-1.0,-1.0,-1.0, 1.0,-1.0,-1.0, 1.0,1.0,-1.0,
-1.0,1.0,-1.0,-1.0,-1.0,1.0,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

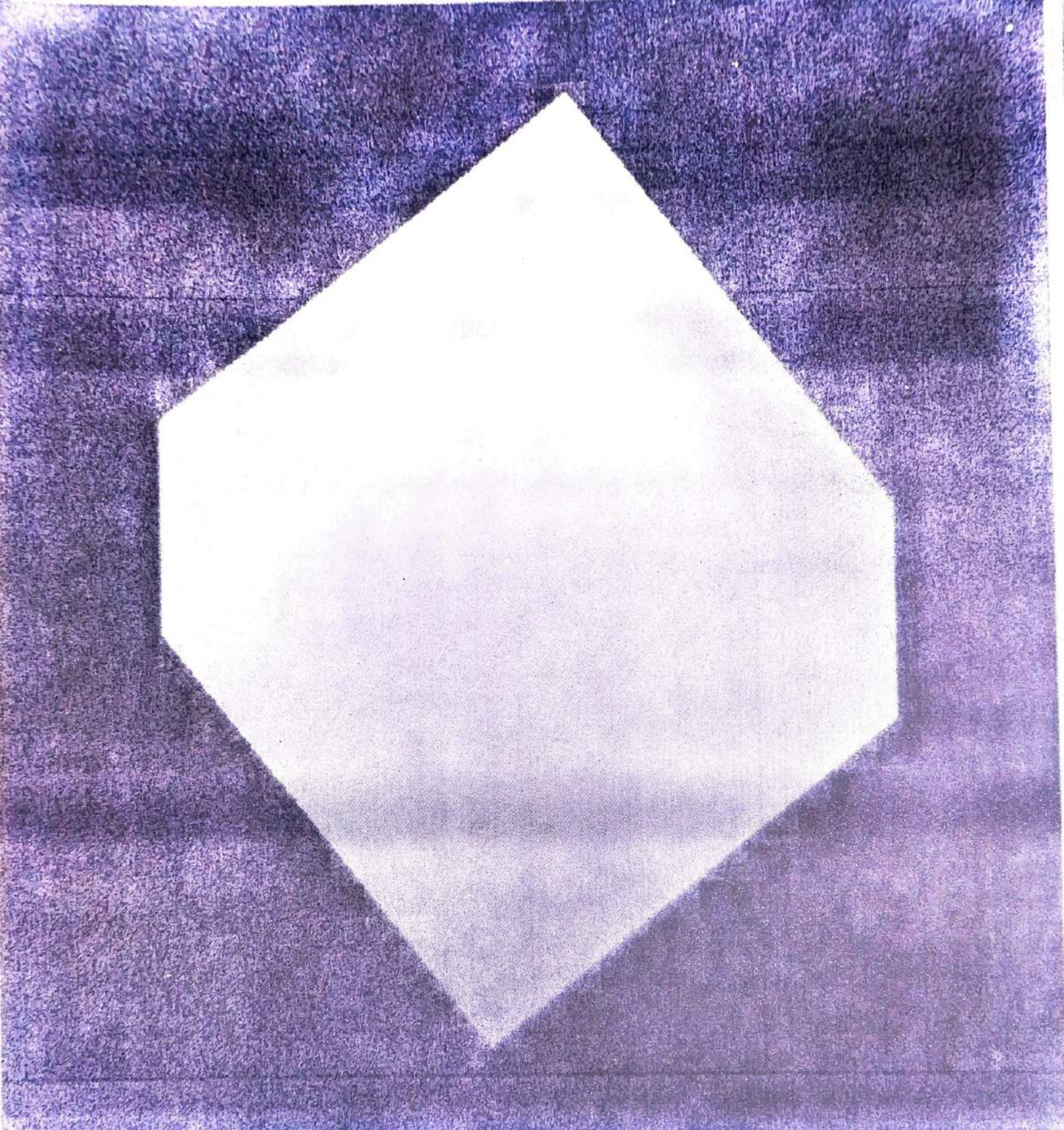
Teacher's Signature:

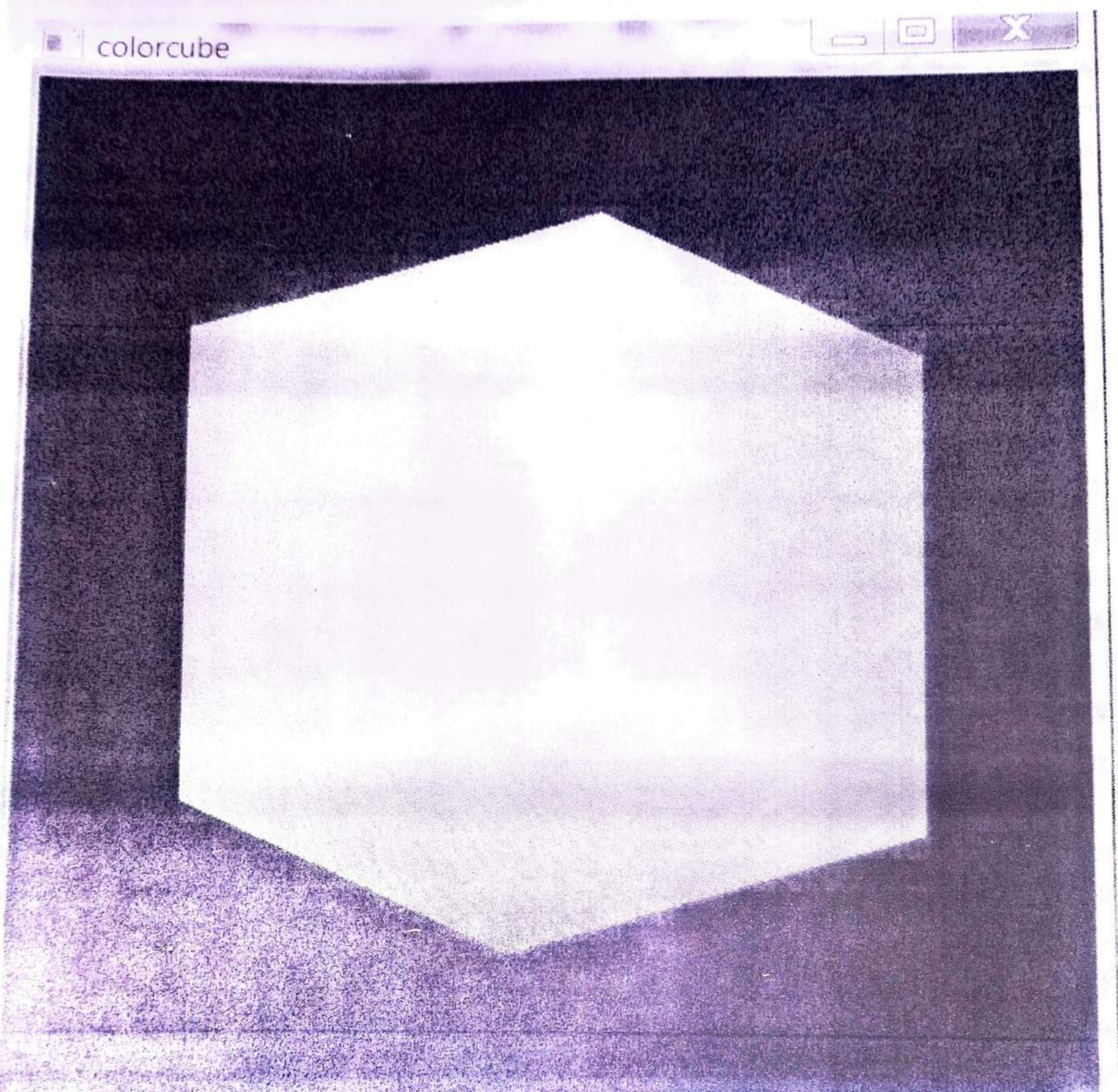
```
void spincube() {
    delay(0.01);
    theta[axis] += 2.0;
if (theta[axis] > 360.0) theta[axis] -= 360.0;
        glutPostRedisplay(); }
void mouse(int btn, int state, int x, int y) {
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) 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);
        glutWindowSize(500, 500);
        glutCreateWindow("Color cube");
```

Teacher's Signature:

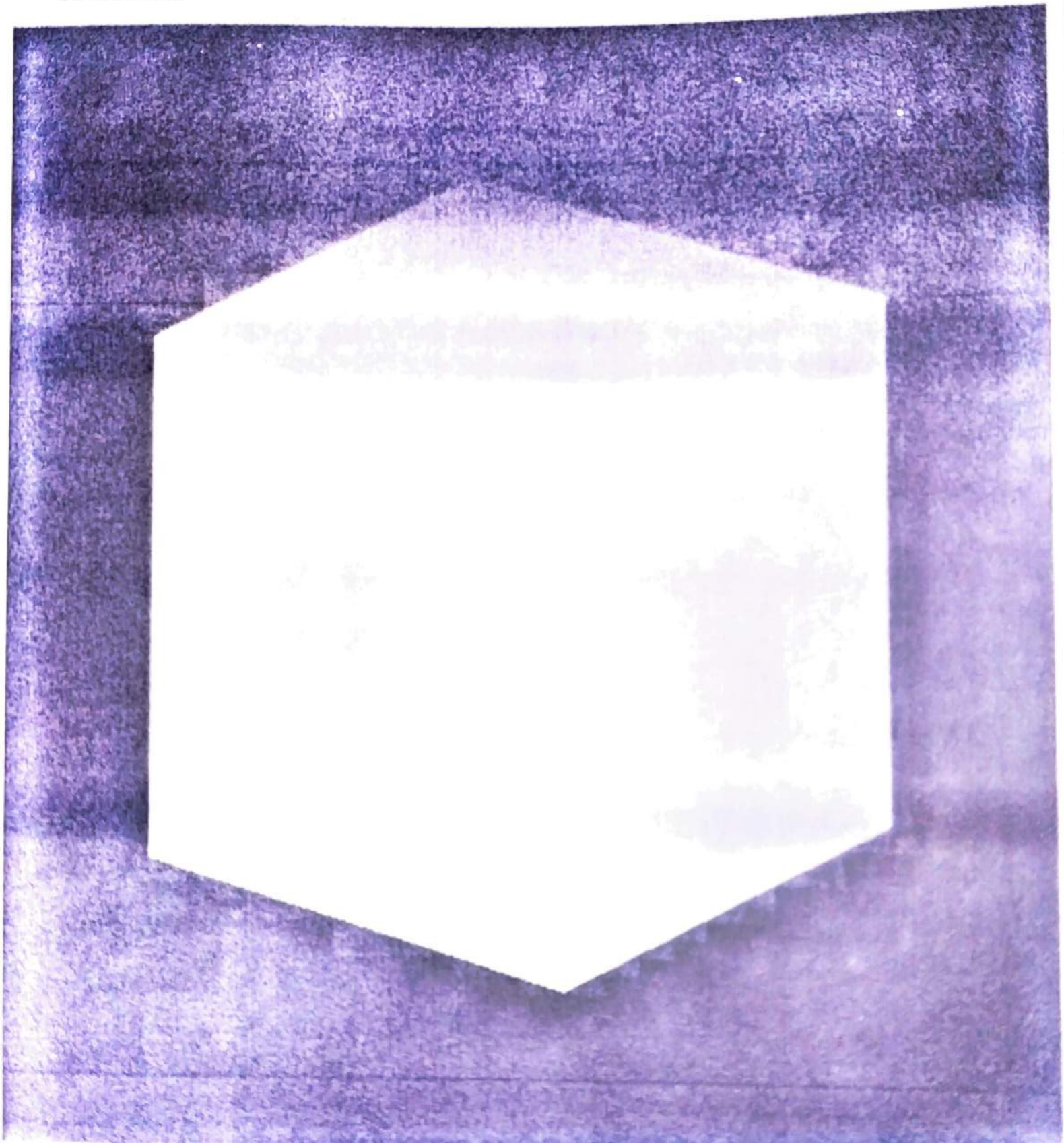
```
glutReshapeFunc(myReshape);
glutDisplayFunc(displaySingle);
glutIdleFunc(spincube);
glutMouseFunc(mouse);
glEnable(GL_DEPTH_TEST);
glEnableClientState(GL_COLOR_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertices);
glPointers(3, GLFLOAT, 0, normals);
glColor3f(1.0, 1.0, 1.0);
glutMainLoop();
}.
```






colorcube

colorcube



Experiment Name / No.: 11 - Curves, colors & Quit.

Camlin | Page No. 37
Date / /

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

```
#include<gl/glut.h>
#include<math.h>
#include<stdio.h>
struct screenPt {
        int x; int y; };
typedef enum{ Limacon=1, Cardioid=2, threeleaf=3, Spiral=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 (screen Pt P1, screenPt P2) {
        glBegin (GL-LNES);
        glVertex2i (P1.x, P1.y);
        glVertex2i (P2.x, P2.y);
        glEnd();
        glFlush(); }.
Void drawCurve (int curveNum) {
```

Teacher's Signature:

```
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 spiral: break;
   default; break; }
theta = dtheta;
while (theta < twoPi) {
      switch (curveNum) {
      case limacon: r = a*cos(theta) + b; break;
      case cardioid: r = a+(1 + cos(theta)); break;
      case 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:

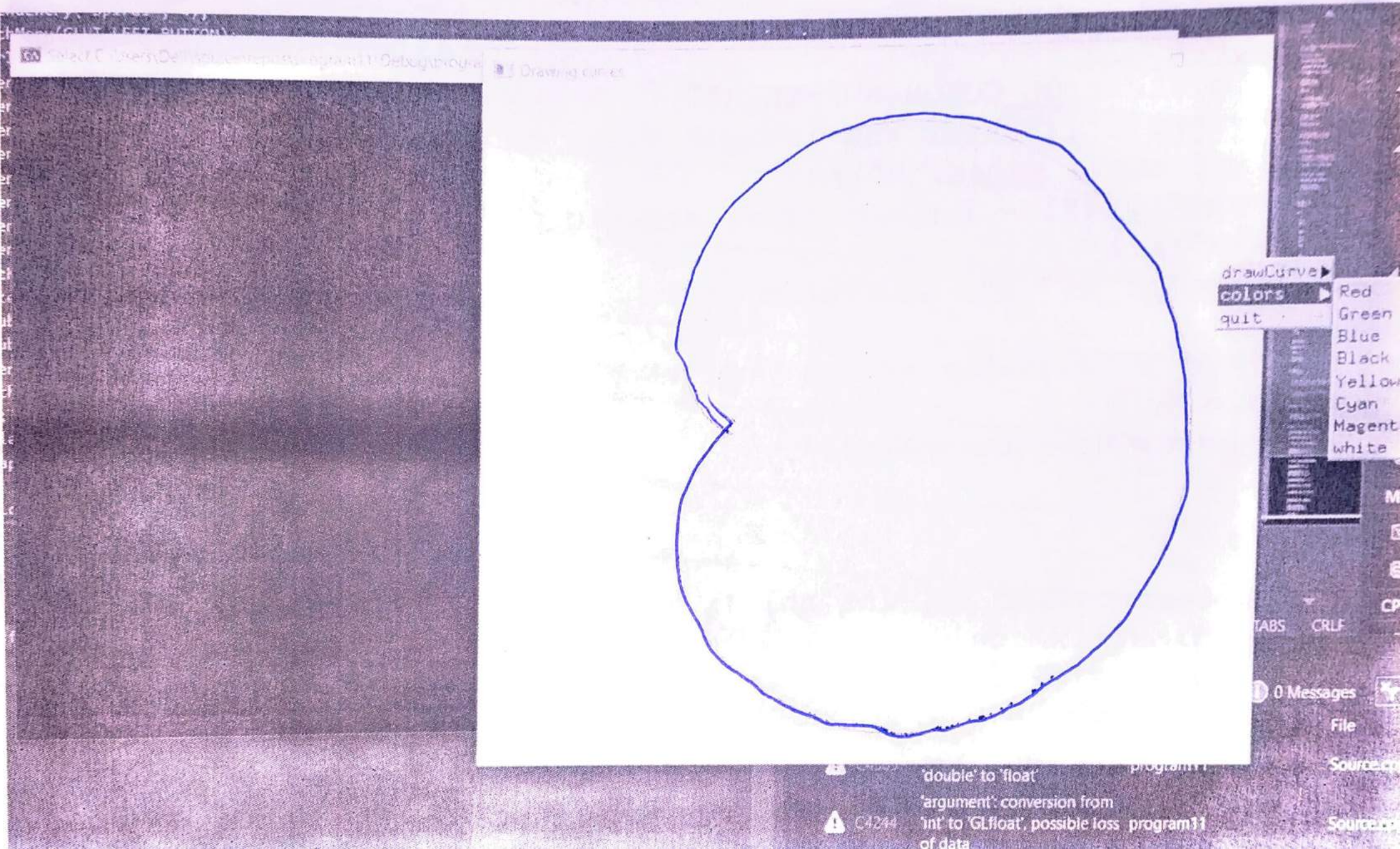
```
void colormenu(int id) {
    switch(id) {
        case 0: break;
        case 1: red=0; green=0; blue=1; break;
        case 2: red=0; green=1; blue=0; break;
        case 3: red=0; green=1; blue=1; break;
        case 5: red=1; green=0; blue=1; break;
        case 6: red=1; green=1; blue=0; break;
        case 7: red=1; green=1; blue=1; break;
        default: break; }
    drawCurve(curve); }
void main_menu(int id) {
        switch(id) {
    case 3: exit(0);
    default: break; } }
void mydisplay() {
        glClear(GL_COLOR_BUFFER_BIT);
        }
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);
```

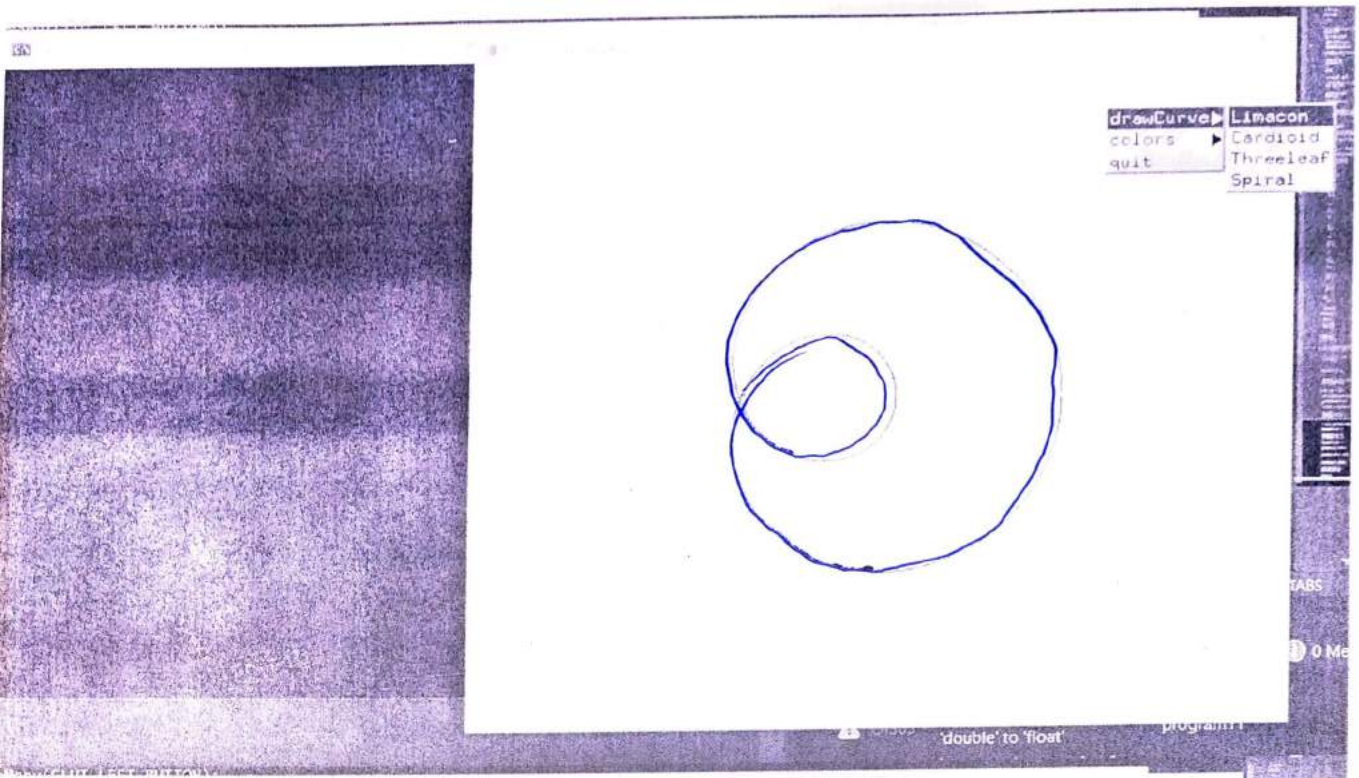
Teacher's Signature:

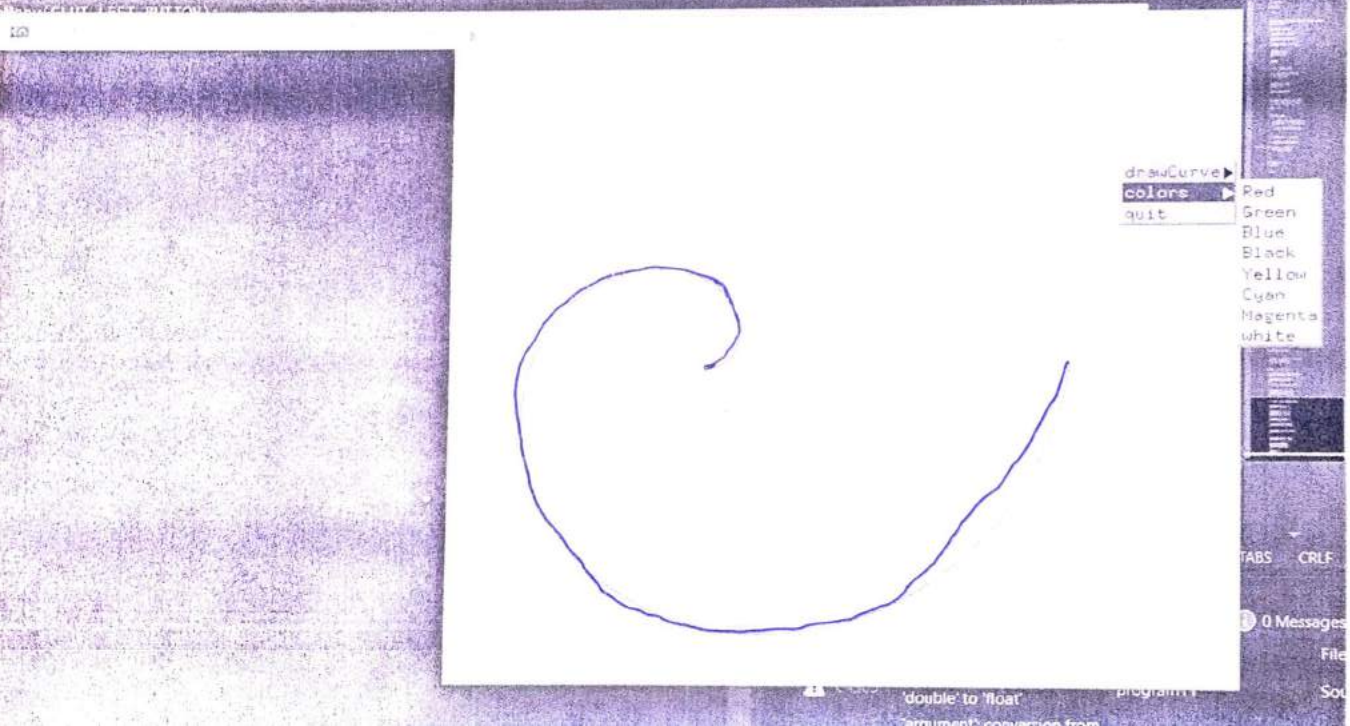
```
glutInitWindowSize (w, h);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Drawing Curves");
int curveId = glutCreateMenu (drawCurve);
glutAddMenuEntry ("Limacon", 1);
glutAddMenuEntry ("Cardioid", 2);
glutAddMenuEntry ("Spiral", 4);
glutAttachMenu (GLUT_LEFT_BUTTON);
int colorId = glutCreateMenu (colorMenu);
   glutAddMenuEntry ("Red", 4);
   glutAddMenuEntry ("Green", 2);
   glutAddMenuEntry ("Black", 0);
   glutAddMenuEntry ("cyan", 3);
   glutAddMenuEntry ("magenta", 5);
   glutAddMenuEntry ("white", 7);
   glutAttachMenu (GLUT_LEFT_BUTTON);
    glutCreateMenu (main-menu);
    glutAddSubMenu ("drawCurves", CurveId);
    glutAddSubMenu ("colors", colorId);
    glutAddMenuEntry ("quit", 3);
    glutAttachMenu (GLUT_LEFT_BUTTON);
      myinit ();
    glutDisplayFunc (mydisplay);
    glutReshapeFunc (myreshape);
      glutMainLoop ();
}.
```

Teacher's Signature:

OUTPUT:


drawCurve
colors
quit
Red
Green
Blue
Black
Yellow
Cyan
Magent
white
TABS
CRLF
0 Messages
File
'double' to 'float'
'argument': conversion from
'int' to 'GLfloat', possible loss
of data
program11


drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral
'double' to 'float'


drawCurve
colors
quit
Red
Green
Blue
Black
Yellow
Cyan
Magenta
White
'double' to 'float'

Experiment Name / No.: 12 - Bezier curve

Camlin / Page No. 41
Date / /

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, xc[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)
          {
```

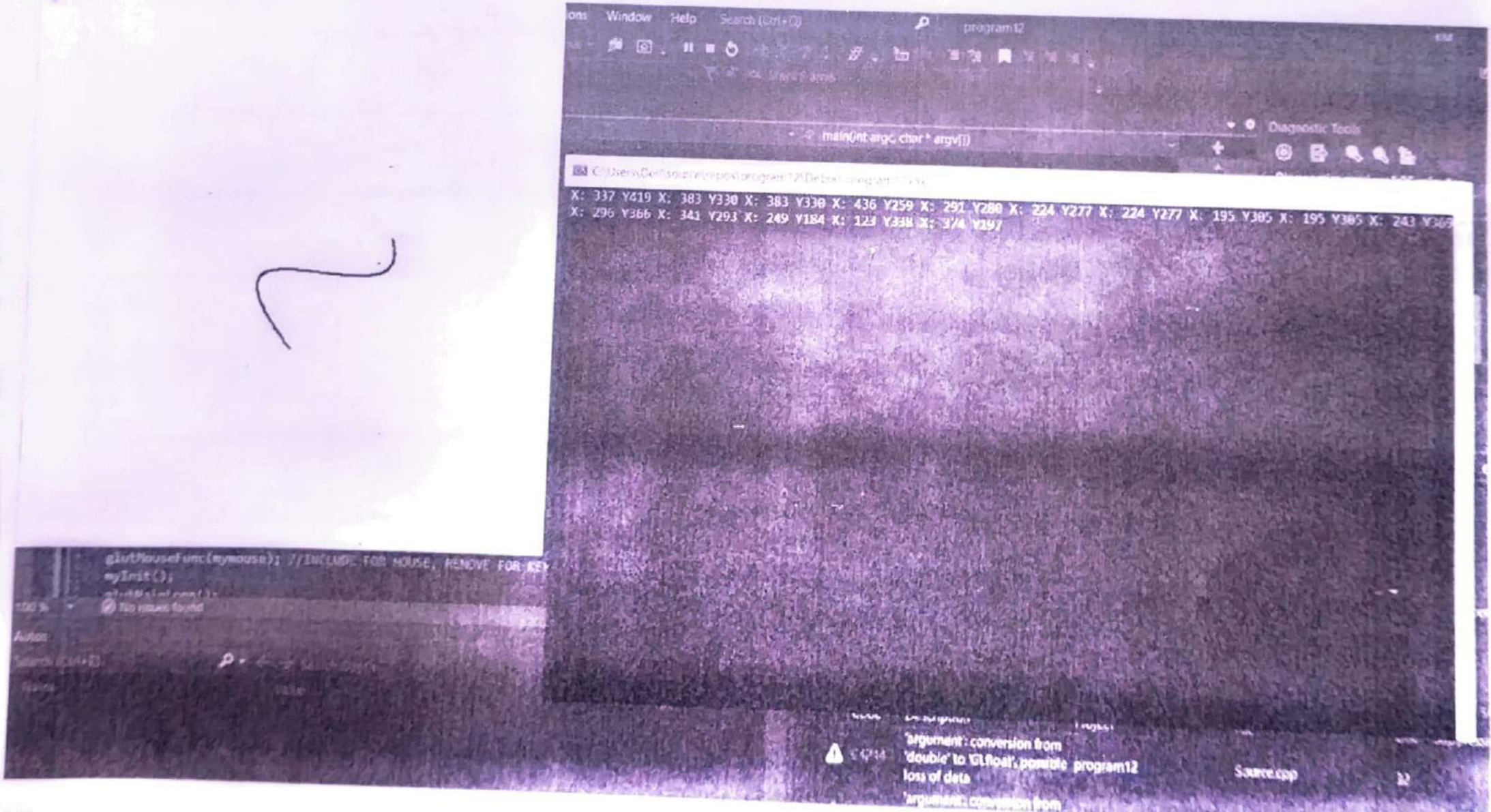
Teacher's Signature:

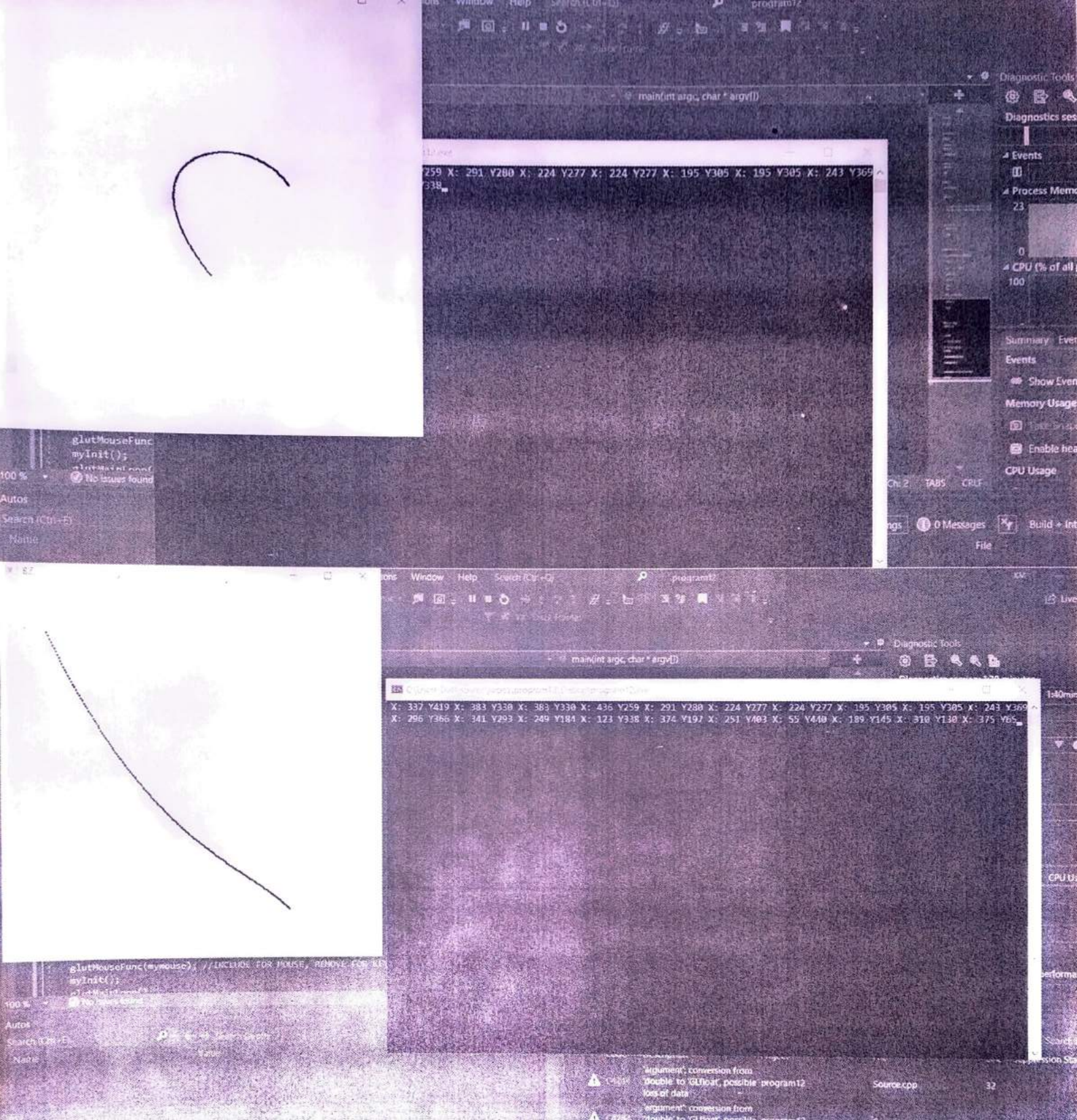
```
double xt= pow(1-t,3)* x1[0]+ 3*t*pow(1-t,2)* x1[1]+
 3* pow(t,2) *(1-t)* x1[2]+pow[t,3)* x1[3];
double yt=pow(1-t,3)*yc[0]+3*t*pow(1-t,2)*pow
   (t,3) * yc[3];
       glVertex2f (xt, yt);
   }
   glcolor3f (1,1,0);
   for(i=0; i<4; i++) {
          glVertex2f (x1[i],yc[i]);
           glEnd();
           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:

```
int main (int argc, char* argv[]) {
    glutInit (&argc, argv);
cout << "Enter the X-co-ordinates" ;
cin >> xc[0] >> xc[1] >> xc[2] >> xc[3];
cout << "enter y co-ordinates" ;
cin >> yc[0] >> yc[1] >> yc[2] >> yc[3];
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (500, 500)
   glutInitWindowPosition (0, 0);
   glutCreateWindow ("B2");
   glutDisplayFunc (display);
   glutMouseFunc (mymouse);
      myInit ();
        glutMainLoop ();
}.
```

OUTPUT:


Window
Help
Search (Ctrl+Q)
program12
Diagnostic Tools
main(int argc, char * argv[])
X: 337 Y419 X: 383 Y330 X: 383 Y330 X: 436 Y259 X: 291 Y280 X: 224 Y277 X: 224 Y277 X: 195 Y305 X: 195 Y305 X: 243 Y365
X: 296 Y366 X: 341 Y293 X: 249 Y184 X: 123 Y338 X: 374 Y297
glutMouseFunc(mymouse); //INCLUDE FOR MOUSE, REMOVE FOR KEY
myInit();
No issues found
'argument': conversion from
'double' to 'GLfloat', possible
loss of data
program12
Source.cpp


259 X: 291 Y280 X: 224 Y277 X: 224 Y277 X: 195 Y305 X: 195 Y305 X: 243 Y369
338
main(int argc, char * argv[])
Diagnostic Tools
Events
Process Memo
CPU (% of all
Summary
Events
Show Even
Memory Usage
Enable hea
CPU Usage
glutMouseFunc
myInit();
No issues found
Autos
Search (Ctrl+E)
Name
TABS
CRLF
0 Messages
Build + Int
File
Window
Help
program12
Diagnostic Tools
main(int argc, char * argv[])
X: 337 Y419 X: 383 Y330 X: 383 Y330 X: 436 Y259 X: 291 Y280 X: 224 Y277 X: 224 Y277 X: 195 Y305 X: 195 Y305 X: 243 Y369
X: 296 Y366 X: 341 Y293 X: 249 Y184 X: 123 Y338 X: 374 Y197 X: 251 Y403 X: 55 Y440 X: 189 Y145 X: 310 Y130 X: 375 Y85
glutMouseFunc(mymouse); //INCLUDE FOR MOUSE, REMOVE FOR KE
myInit();
100 %
No issues found
Autos
Search (Ctrl+E)
Name
argument: conversion from
double to GLfloat, possible
loss of data
program12
Source.cpp
32