Certificate

Name: Pavan kumar Y

Class: 4th year 7th Sem

Roll No: 1RV18CS421

Exam No:

Institution R.V. College of Engineering

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

No. of practicals certified 12 *out of* 12 *in the subject of* Computer Graphics Lab

.....
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	write a Pgm to generate a line using Bresenham's line drawing technique	1-4	5/10/20		
2	write a Pgm to generate a circle & ellipse using Bresenham's circle drawing & ellipse drawing technique	5-11	12/10/20		
3	write a Pgm to recursive subdivides a tetrahedron to form 3D Sierpinski gasket	12-14	19/10/20		
4	write a Pgm to fill any given polygon using Scan line area filling algorithm	15-17	2/11/20		
5	write a Pgm to Create a house like figure & Perform the following Operation: • Rotate it • Reflect it	18-21	9/11/20		
6	write a pgm to implement the Cohen-Sutherland line clipping algorithm	22-26	23/11/20		

Index

Date 5/10/20

Expt. No. 1

Page No. 1

write a pgm 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 & specify input through keyboard/mouse.

```
#include <iostream>
#include <GL/glut.h>
#include <time.h>
using namespace std;
int x1, x2, yc1, y2, 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 - yc1;
    if (dx < 0) dx = -dx;
    if (dy < 0) dy = -dy;
      incx = 1;
     if (y x2 < x1)
        inc1 = -1;
```

Teacher's Signature

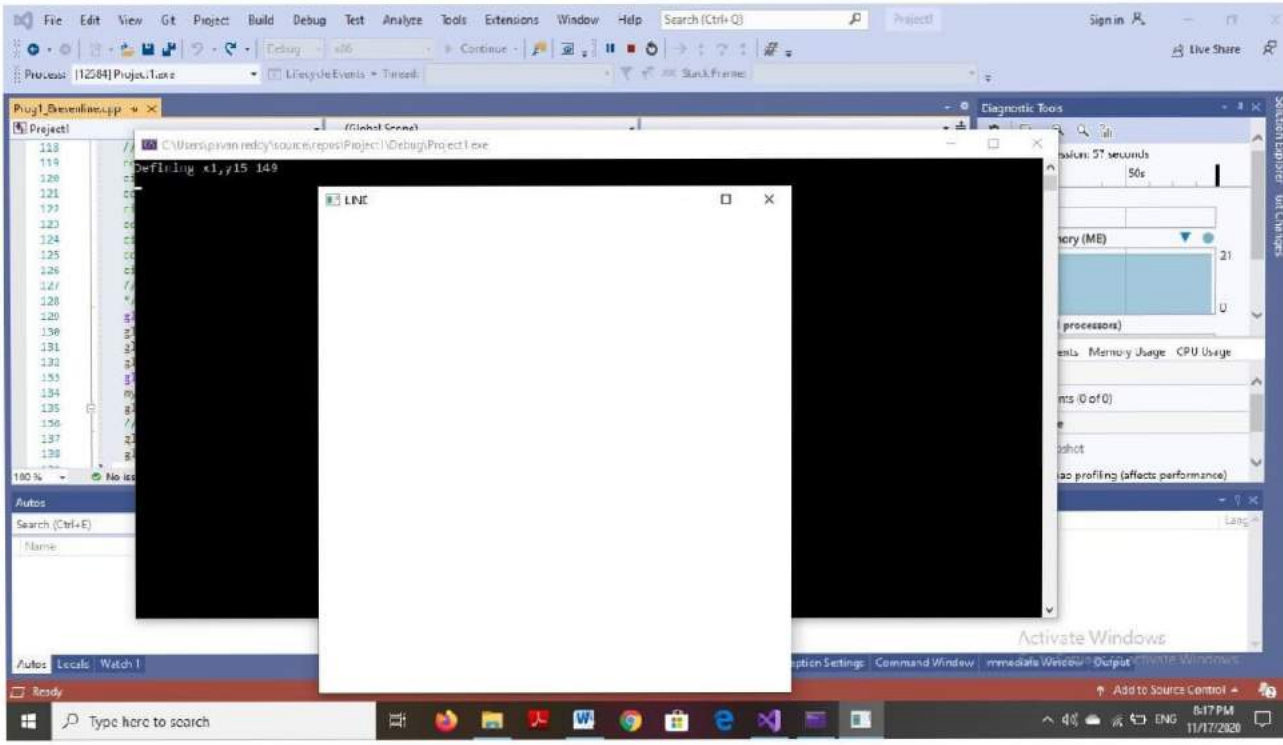
```
incy=+1;
if(y2<y1)
  incy=-1;
x=x1;
y=y1;
if(dx>dy)
{
   drawpixel(x,y);
   e=2*dy-dx;
   inc1=2*(dy-dx);
   inc2=2*dy;
   for(i=0;i<dx;i++)
   {
      if(e>0)
      {  y+=incy;
         e+=inc+1;
      }
      else
         e+=inc2;
      x+=incx;
      draw_pixel(x,y);
   }
}
else {
   draw-pixel(x,y);
   e=2*dy-dx;
   inc1=2*(dx-dy);
   inc2=2*dx;
   for(i=0;i<dy;i++){
```

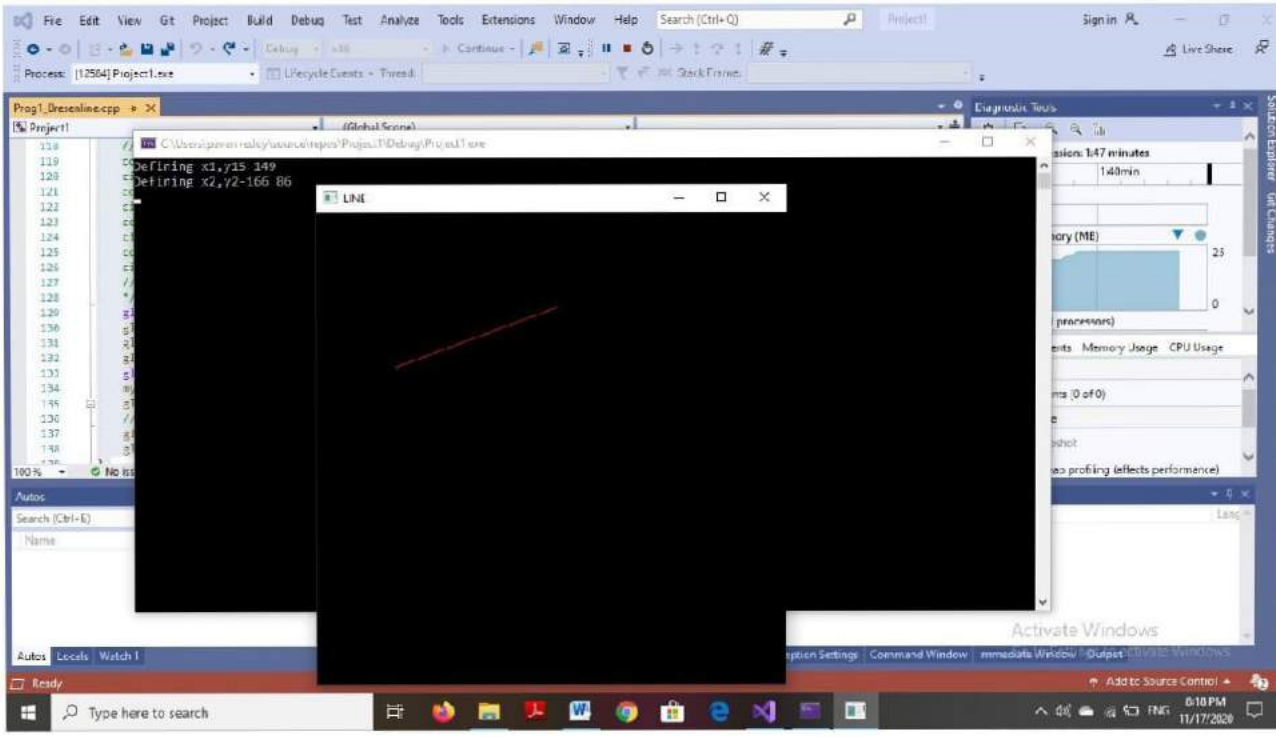
```
        if (e>0)
        {
            x+=inc(x);
            e+=inc1;
        }
        else
            e+=inc2;
        y+=incy;
        draw_pixel(x,y);
    }
}
glFlush();
}
void myinit()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glClearColor(1,1,1,1);
  glOrtho2D(-250,250,-250,250);
}
void mymouse(int button, int state, int x, int Y)
{
   switch(button)
   {
     case GLUT_LEFT_BUTTON:
         if(state==GLUT_DOWN)
         {
           if(flag==0)
           {
             Printf("Defining X1,Y1");
```

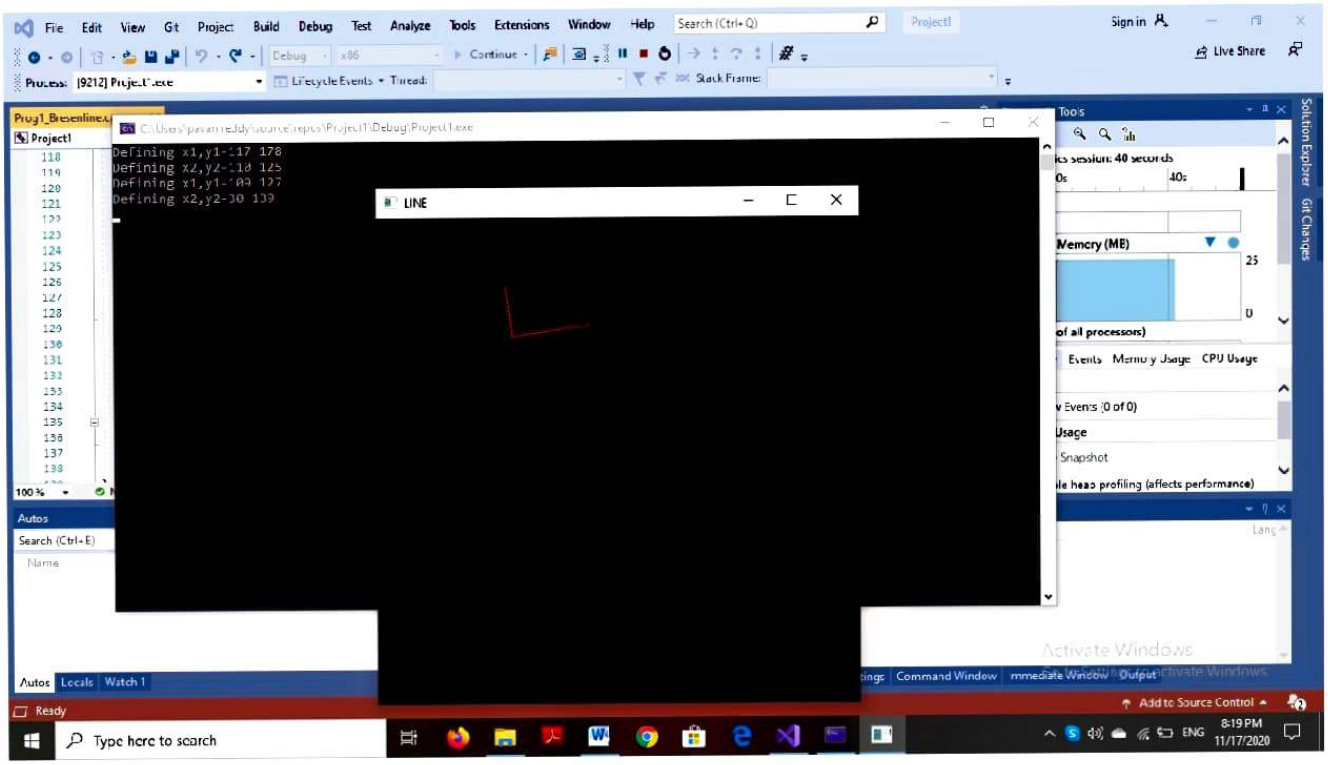
Teacher's Signature

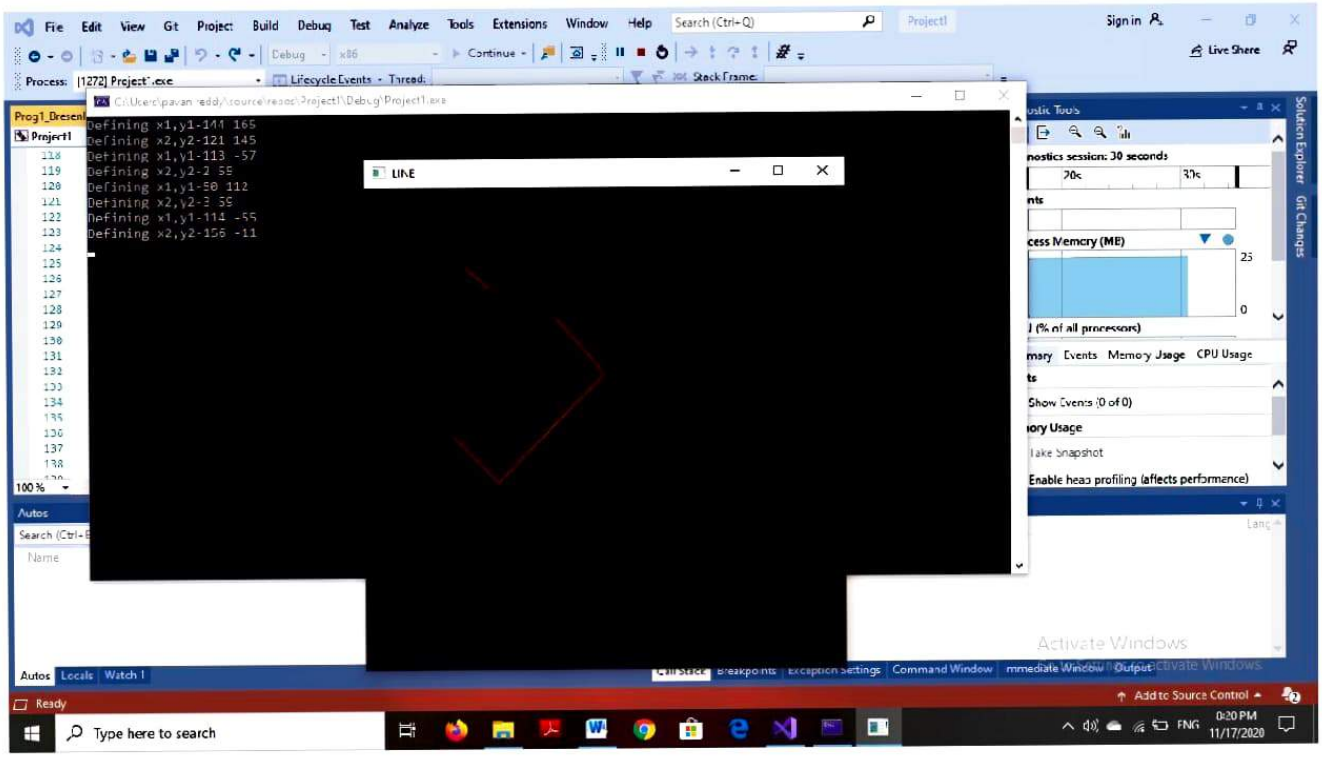
```
            x1 = x-250,
            yc1 = 250-y;
            flag++;
            cout << x1 << " " << yc1 << "\n";
        }
        else
        {
          Printf ("Defining x2, y2");
            x2 = x-250;
            y2 = 250-y;
            flag = 0;
            cout << x2 << " " << y2 << "\n";
            draw_line();
        }
    }
    break;
  }
}
void display() { }
 int main (int ac, char** av[])
  {
    glutInit(&ac, av);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glut InitWindowPosition (100, 200);
    glutCreateWindow ("line");
    myinit();
    glutMouseFunc (mymouse);
    glutDisplayFunc (display);
    glutMainLoop();
}
```

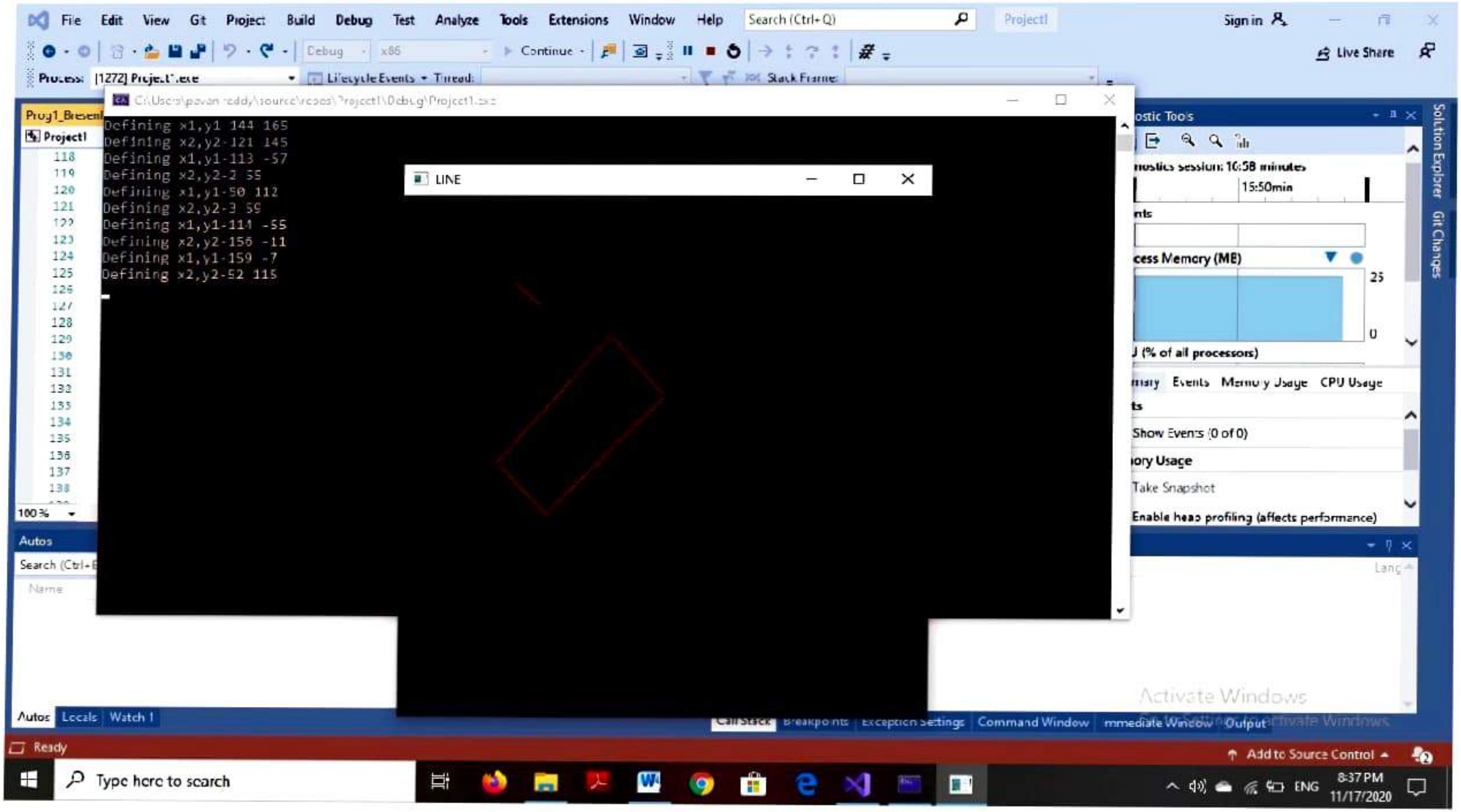
Teacher's Signature


Defining x1,y15 149
LINE


Defining x1,y15 149
Defining x2,y2-166 86
LINE


Defining x1,y1-117 178
Defining x2,y2-110 125
Defining x1,y1-89 137
Defining x2,y2-30 139
LINE


Defining x1,y1-144 165
Defining x2,y2-121 145
Defining x1,y1-113 -57
Defining x2,y2-2 58
Defining x1,y1-50 112
Defining x2,y2-3 58
Defining x1,y1-114 -55
Defining x2,y2-156 -11
LINE


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Sign in
Live Share
Debug
x86
Continue
Defining x1,y1 144 165
Defining x2,y2-121 145
Defining x1,y1-113 -57
Defining x2,y2-2 55
Defining x1,y1-50 112
Defining x2,y2-3 59
Defining x1,y1-114 -55
Defining x2,y2-156 -11
Defining x1,y1-159 -7
Defining x2,y2-52 115
LINE
Diagnostic Tools
Solution Explorer
Git Changes
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
Autos
Locals
Watch 1
Command Window
Immediate Window
Output
Ready
Add to Source Control
Type here to search
ENG
8:37 PM
11/17/2020

Date 12/10/20

Expt. No. 2 Page No. 5

write a pgm to generate a circle & ellipse using Bresenham's circle drawing & ellipse drawing technique. use 2 windows to draw circle in one window & ellipse in the other window. user can specify input 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_COLOUR_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 mymouse(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 = Pl.x;
    yc = Pl.y;
    float exp = (P2.x - Pl.x) * (P2.x - Pl.x) + (P2.y -
          Pl.y) * (P2.y - Pl.y);
    r = (int)(sqrt(exp));
    Circlebres();
    Part_done = 0;
  }
}
void draw_ellipse(int xce, int yce, int x, int y)
{
    glBegin(GL_POINTS);
    glVertex(x+xce, y+yce);
    glVertex(-x+xce, y+yce);
    glVertex(x+xce, -y+yce);
    glVertex(-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--;
      dx = dx + (2 * ry * ry);
      dy = dy - (2 * rx * rx);
      d1 = d1 + dx - dy + (ry * ry);
   }
}
d2 = ((ry*ry) * ((x+0.5) * (x+0.5))) + ((rx *
        rx) * ((y-1) + (y-1))) - (rx*rx*ry*ry);
while (y >=0) {
   draw_ellipse (xce, yce, x, y);
    if (d2 >0) {
       y--;
       dy = dy - (2 * rx * rx);
       d2 = d2 + (rx * rx) - dy;
    }
    else {
       y--;
       x++;
       dx = dx + (2 * ry * ry);
       dy = dy - (2 * rx * rx);
       d2 = d2 + dx - dy + (rx * rx);
}}
```

Teacher's Signature _____

```
    glFlush();
}
int p1e_x, p2e_x, p1e_y, p2e_y, p3e_x, p3e_y;
int pointle_done = 0;
void mymouse1(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;
    }
    else if(button == GLUT_LEFT_BUTTON && state == GLUT_
       DOWN && pointle_done == 1) {
        p2e_x = x-250;
        p2e_y = y-250-y;
        float exp = (p2e_x - p1e_x)*(p2e_x - p1e_x)
            + (p2e_y - p1e_y)*(p2e_y - p1e_y);
        rx = (int)(sqrt(exp));
        pointle_done = 2;
    }
    else if (button == GLUT_LEFT_BUTTON && state
       == GLUT_DOWN && pointle_done == 2)
    {
        p3e_x = x-250;
        p3e_y = 250-y;
```

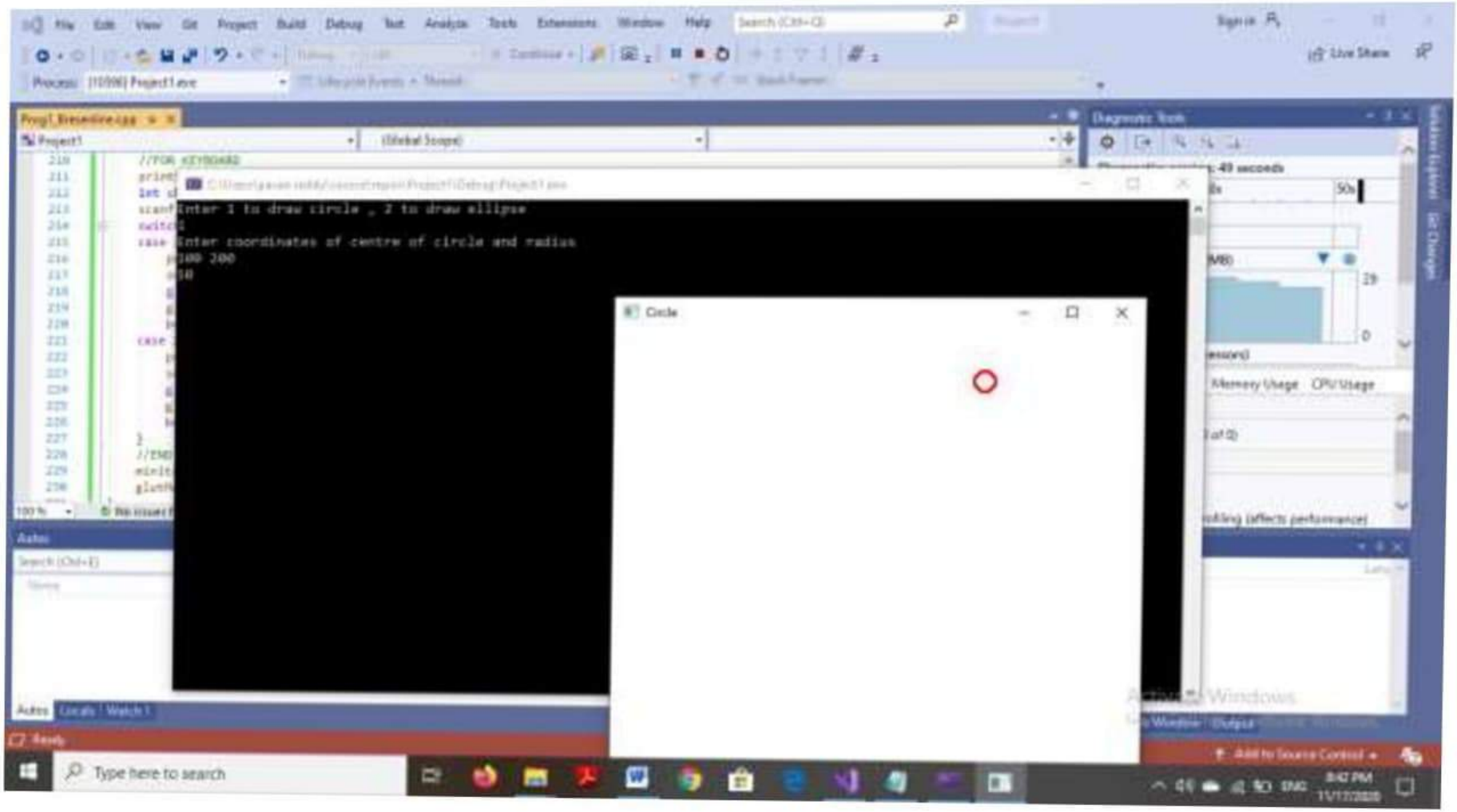
Teacher's Signature

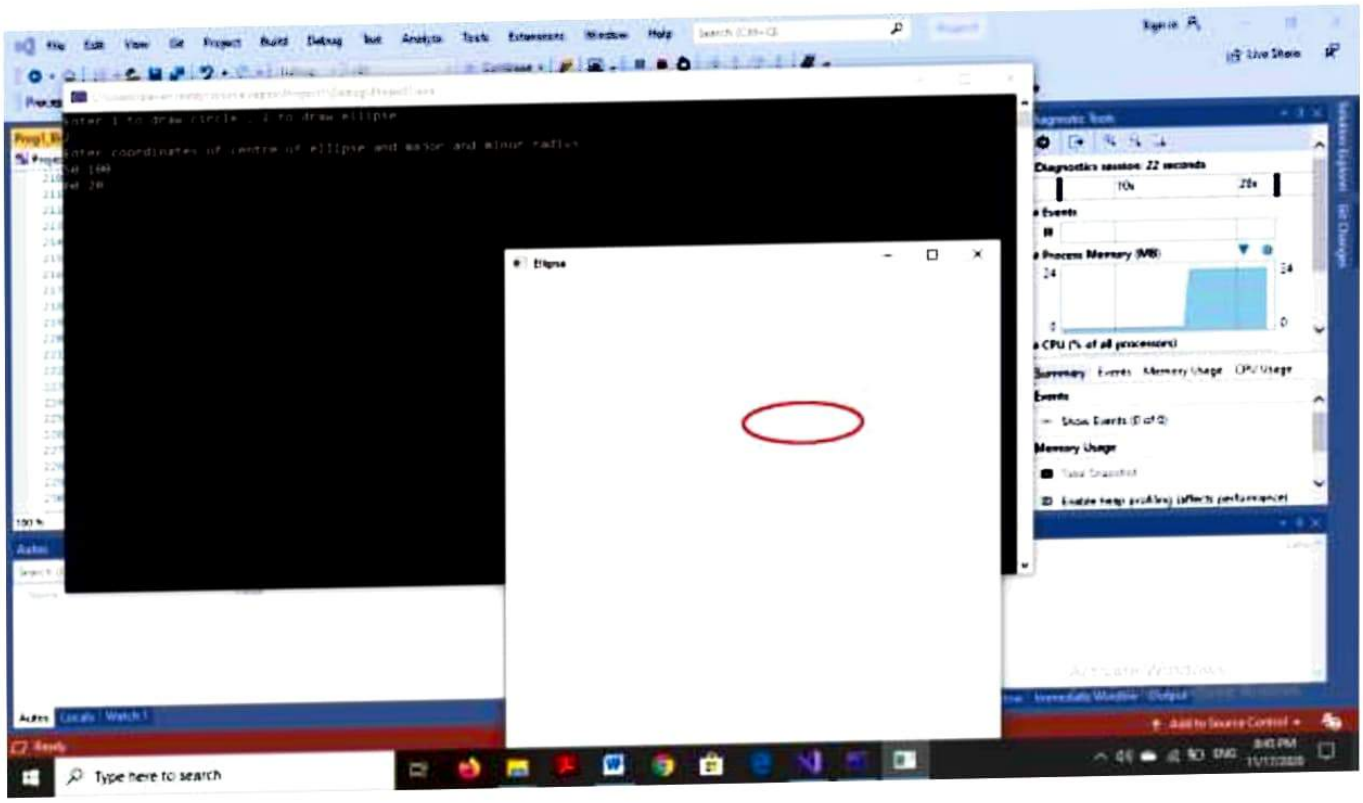
```
        float exp=(p3e_x-p1e_x)*(p3e_x-p1e_x)+(p3e_y
                -p1e_y)+(p3e_y-p1e_y);
        ry=(int)(sqrt(exp));
        midpteellipse();
        pointle_done=0;
    }
}
void mydrawing(){}
void mydrawing(){}
void myinit()
{
    glClearcolor(1,1,1,1);
    glColor3f(1.0,0.0,0.0);
    glPointSize(3.0);
    gluOrtho2D(-250,250,-250,250);
}

void main(int argc, char* argv[])
{
    glutinit(&agc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitwindowSize(500,500);
    glutInitwindowPosition(0,0);
    int id1=glutCreatewindow("Circle");
    glutSetwindow(id1);
    glutmouseFunc(mymouse);
    glutDisplayFunc(myDrawing);
    myinit();
    glutInitwindowSize(500,500);
```

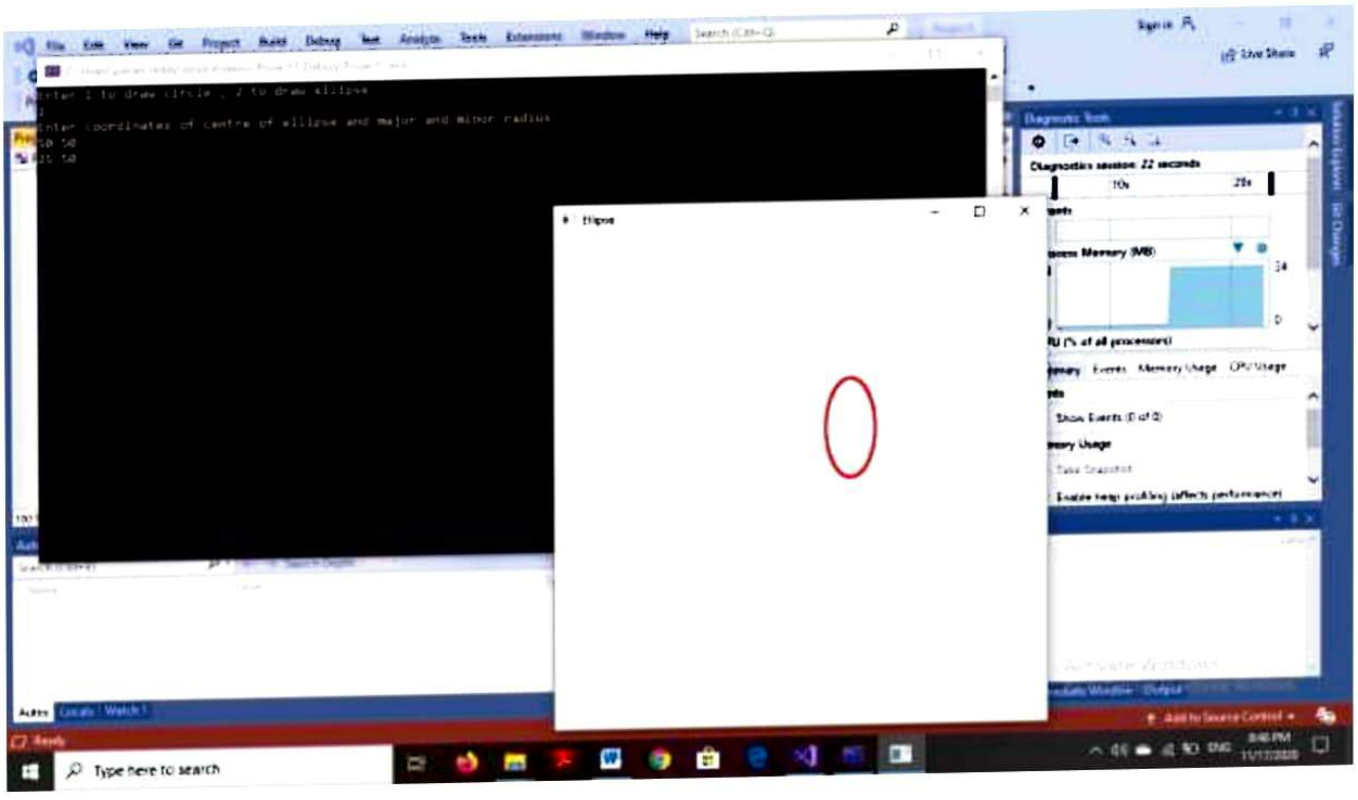
Teacher's Signature

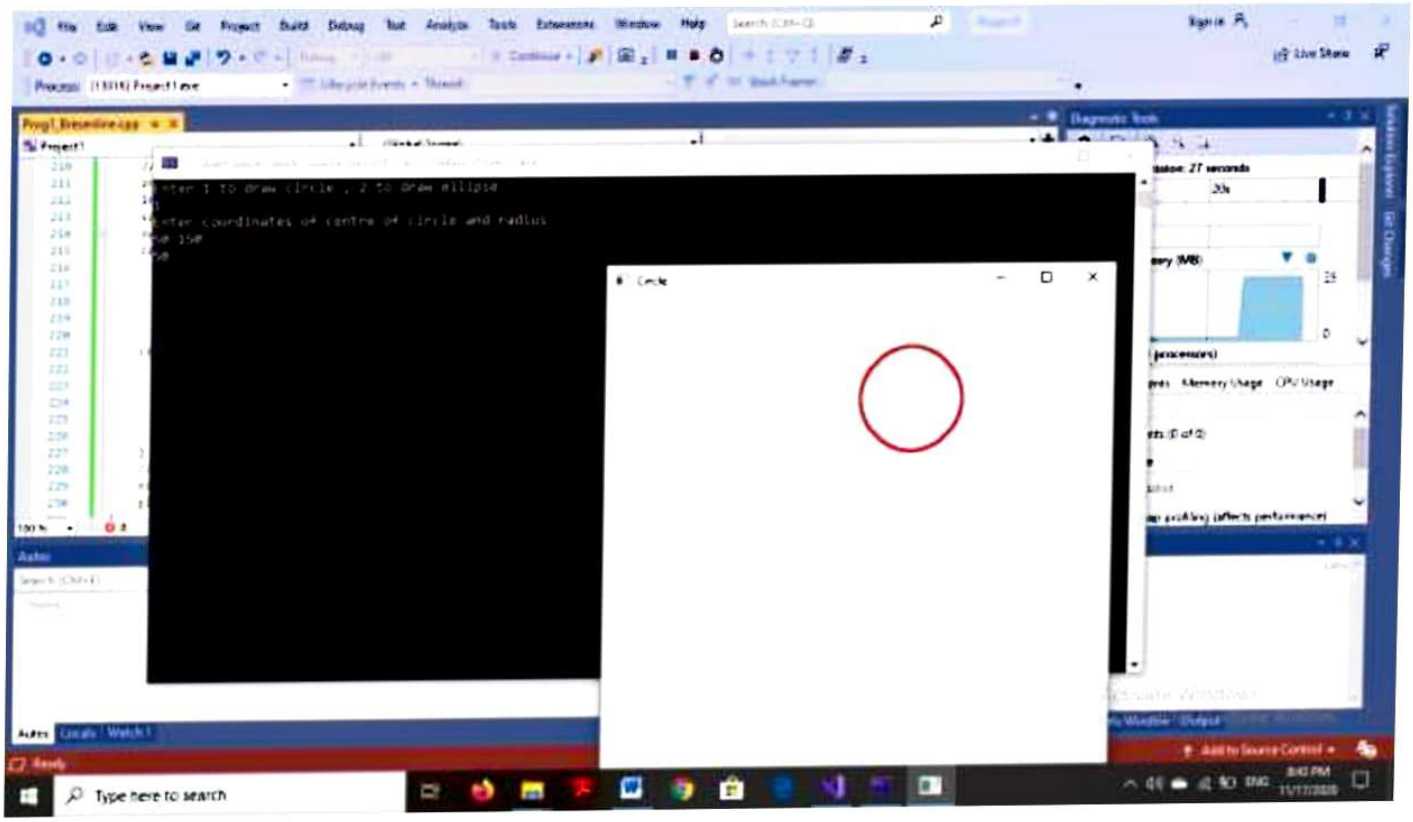
```
glutInitWindowPosition(600,100);
int id2 = glutCreateWindow ("Ellipse");
glutSetWindow(id2);
glutMouseFunc(mymouse1);
glutDisplayFunc(myDrawing);
printf ("Enter 1 to draw circle, 2 to draw
                Ellipse \n");
int ch;
scanf_s("%d", &ch);
switch (ch) {
case 1:
    printf("Enter coordinates of Centre of Circle &
                radius \n");
    scanf_s("%d %d %d", &xc, &yc, &r);
    glutCreateWindow ("Circle");
    glutDisplayFunc (Circlebres);
    break;
case 2:
    printf(" Enter coordinates of centre of ellipse &
            major & minor radius \n");
    scanf_s("%d %d %d %d", &xce, &yce, &rx, &y);
    glutDisplayFunc(midpEllipse);
    break;
}
myinit();
glutMainLoop();
}
```

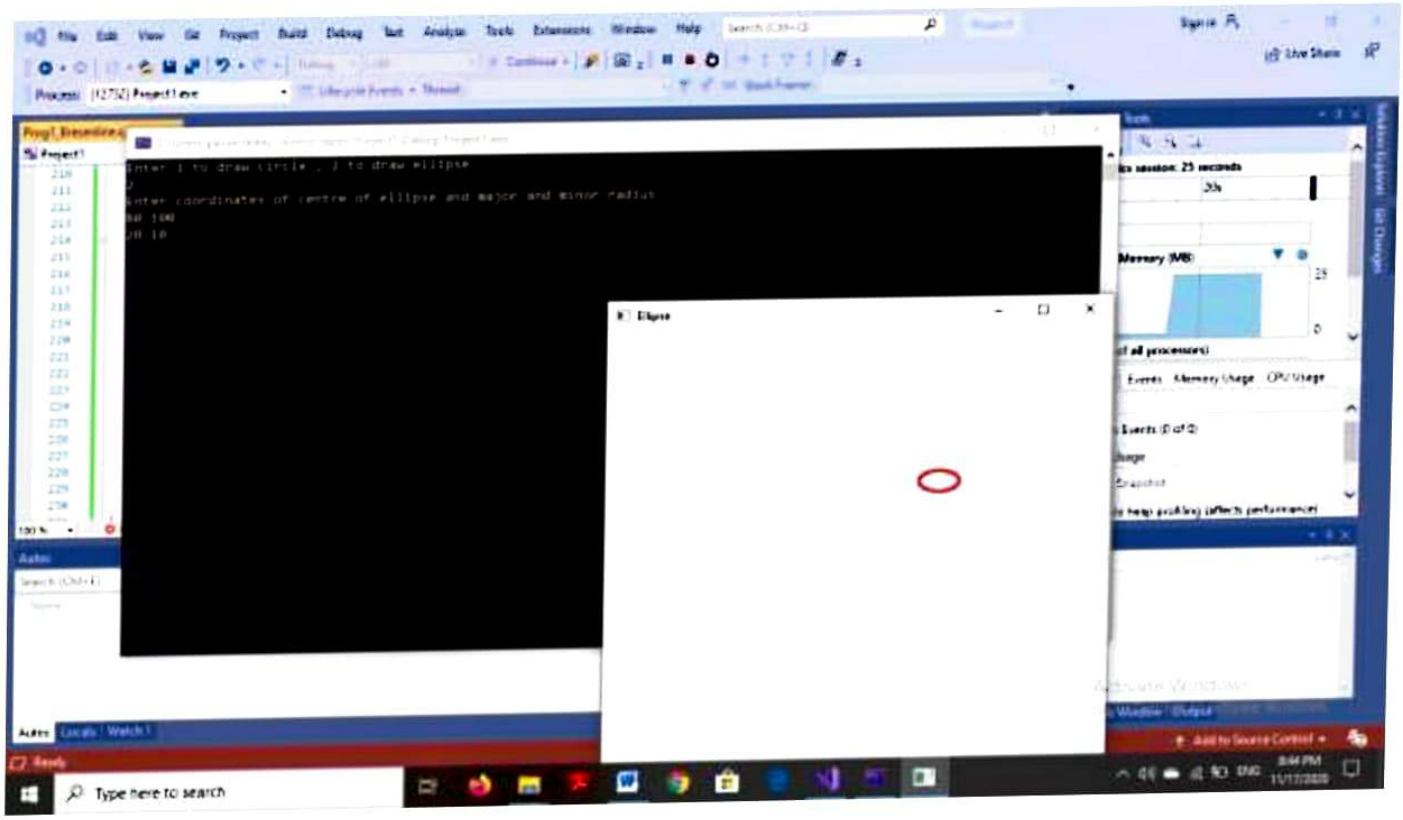
Output:


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


Ellipse
Type here to search


Ellipse
Type here to search


Circle
Type here to search


Ellipse
Type here to search

Date 19/10/20

Expt. No. 3

Page No. 12

write a pgm to recursive step 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 m;
typedef float Point[3];
point tetra[4]={{0,100,-100},{0,0,100},{100,-100,-100},
        {-100,-100,-100}};
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(0.0,0.0,1.0);
  divide_triangle(tetra[0],tetra[3],tetra[1],m);
  glColor3f(0.0,0.0,0.0);
  divide_triangle(tetra[0],tetra[2],tetra[3],m);
  glFlush();
}

void draw_triangle(Point P1, Point P2, Point P3)
{
  glBegin(GL_TRIANGLES);
  glVertex3fv(P1);
  glVertex3fv(P2);
```

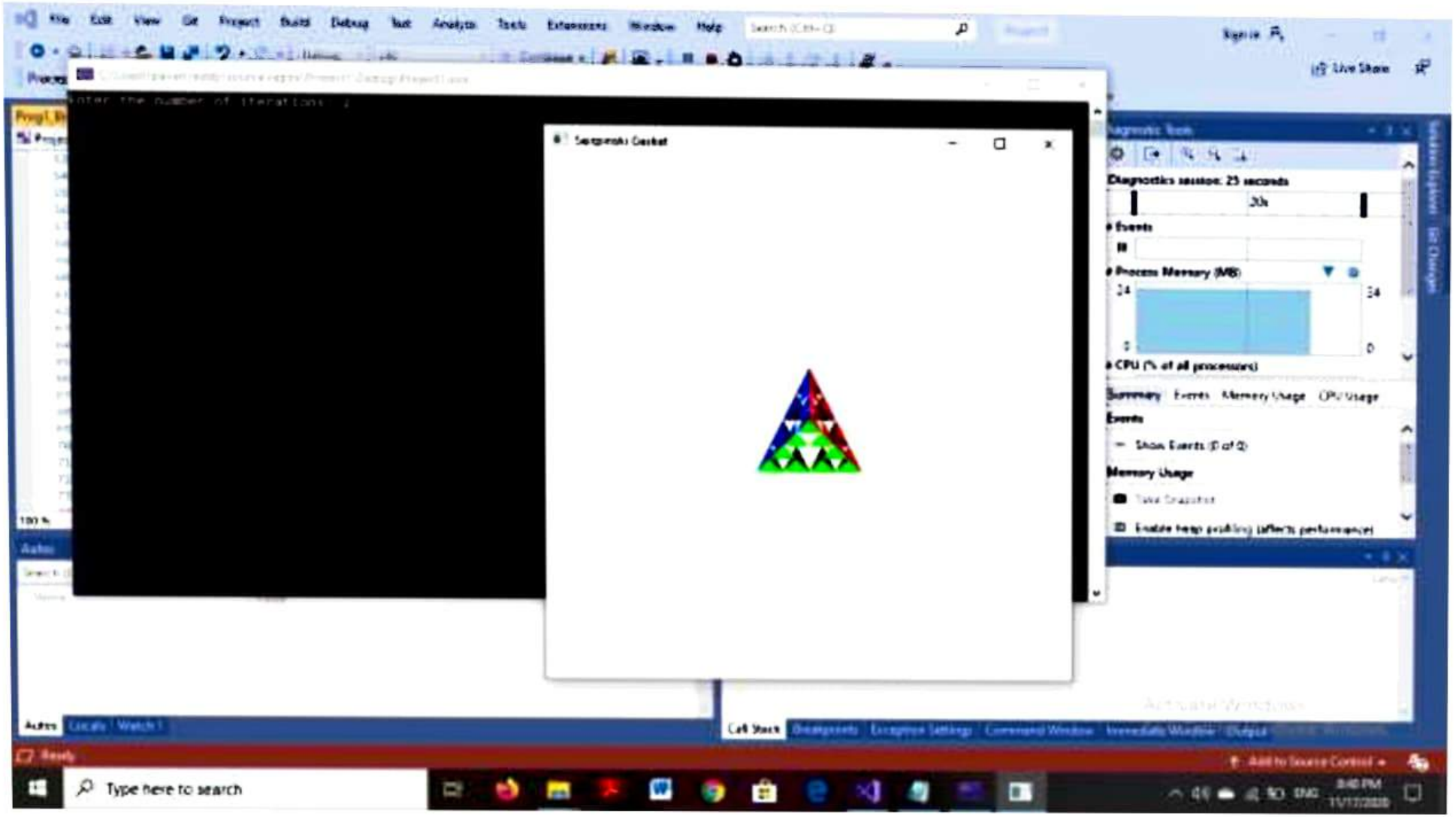
Teacher's Signature

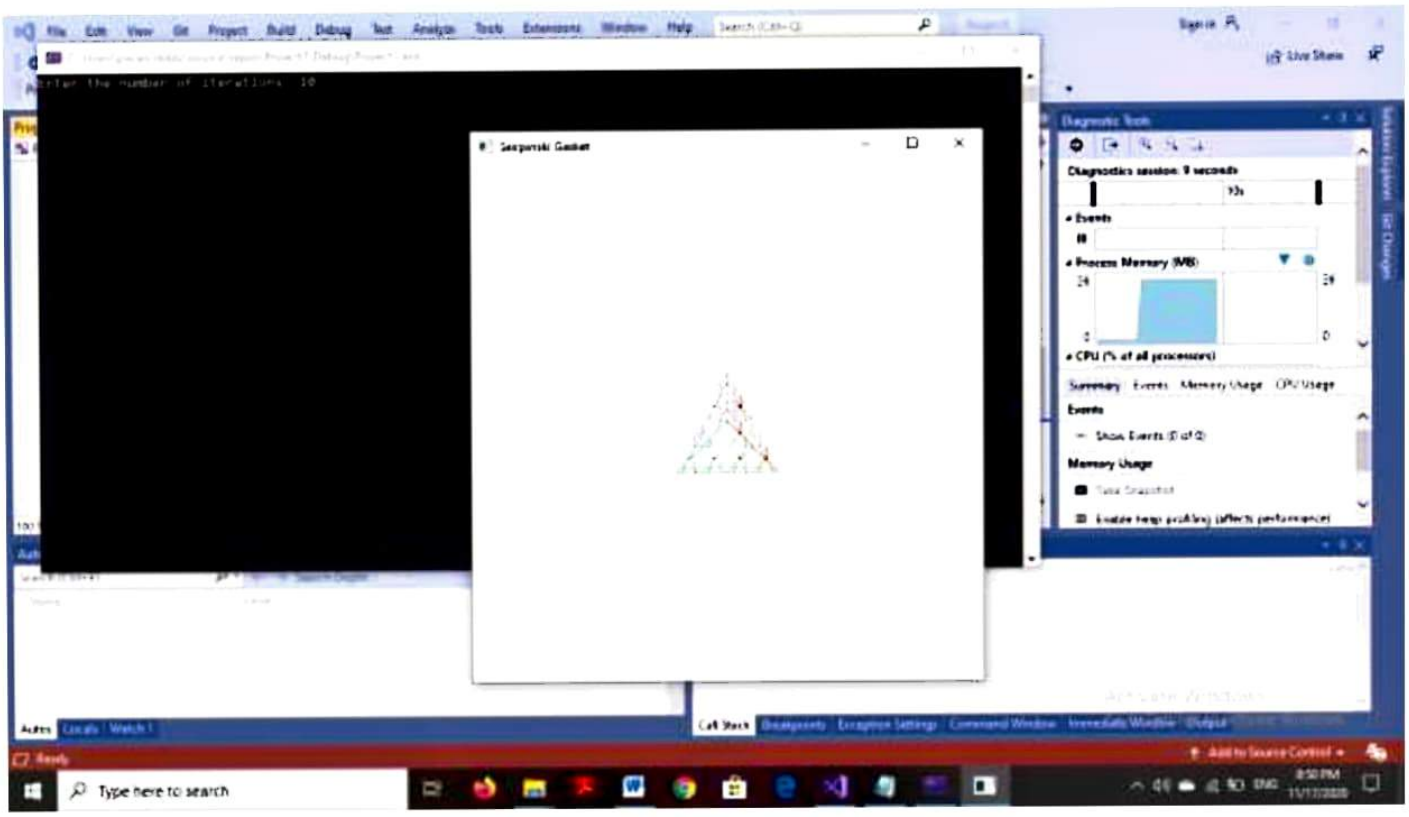
```
glVertex3fv(P3);
glEnd();
}

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, 500.0);
}
```

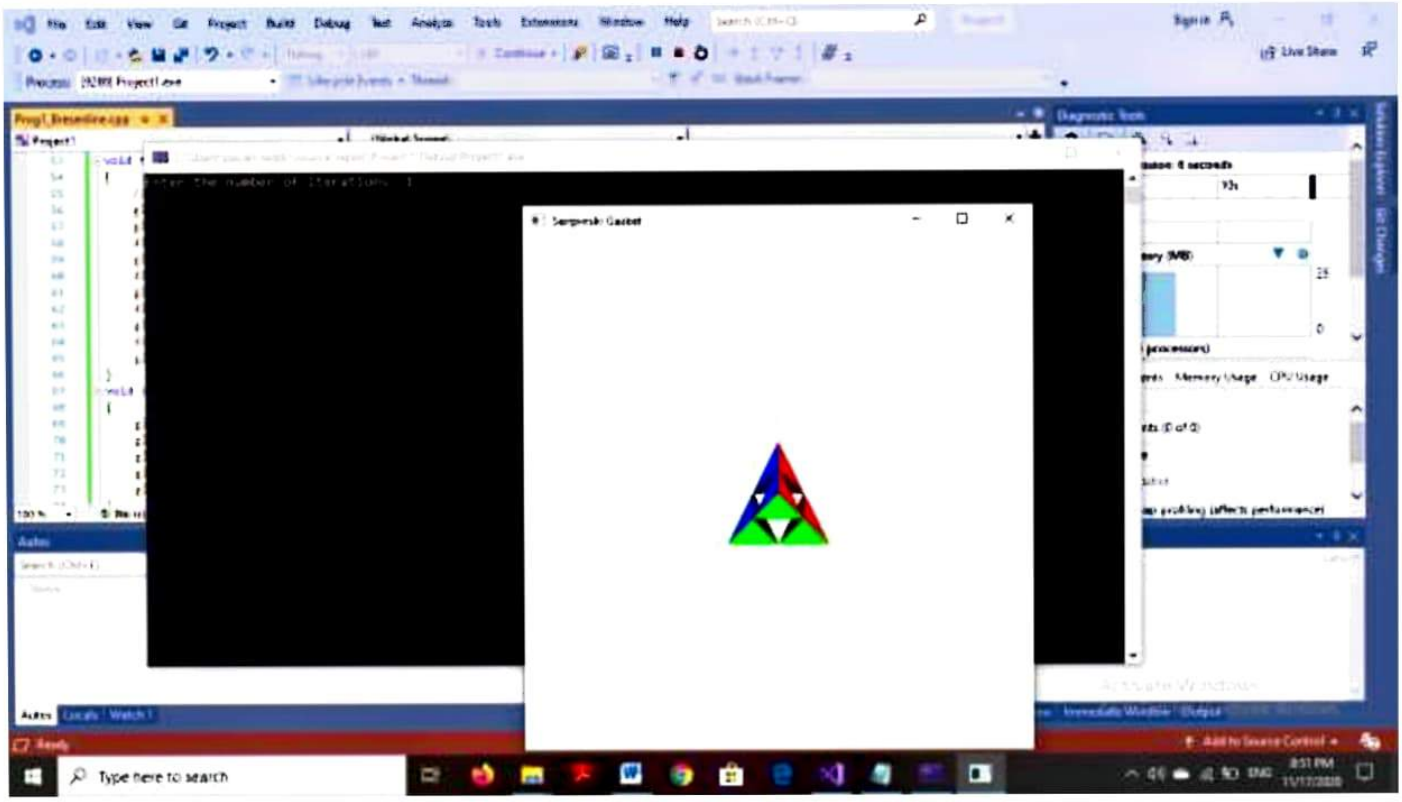
```
int main(int argv, char* argc)
{
    printf("Enter the no of iterations : ");
    scanf_s("%d", &m);
    glutInit(&argv, argc);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB |
                        GLUT_DEPTH);
    glutInitWindowPosition(100, 200);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Seirpinski Gasket");
    glutDisplayFunc(tetrahedron);
    glEnable(GL_DEPTH_TEST);
    myInit();
    glutMainLoop();
}
```

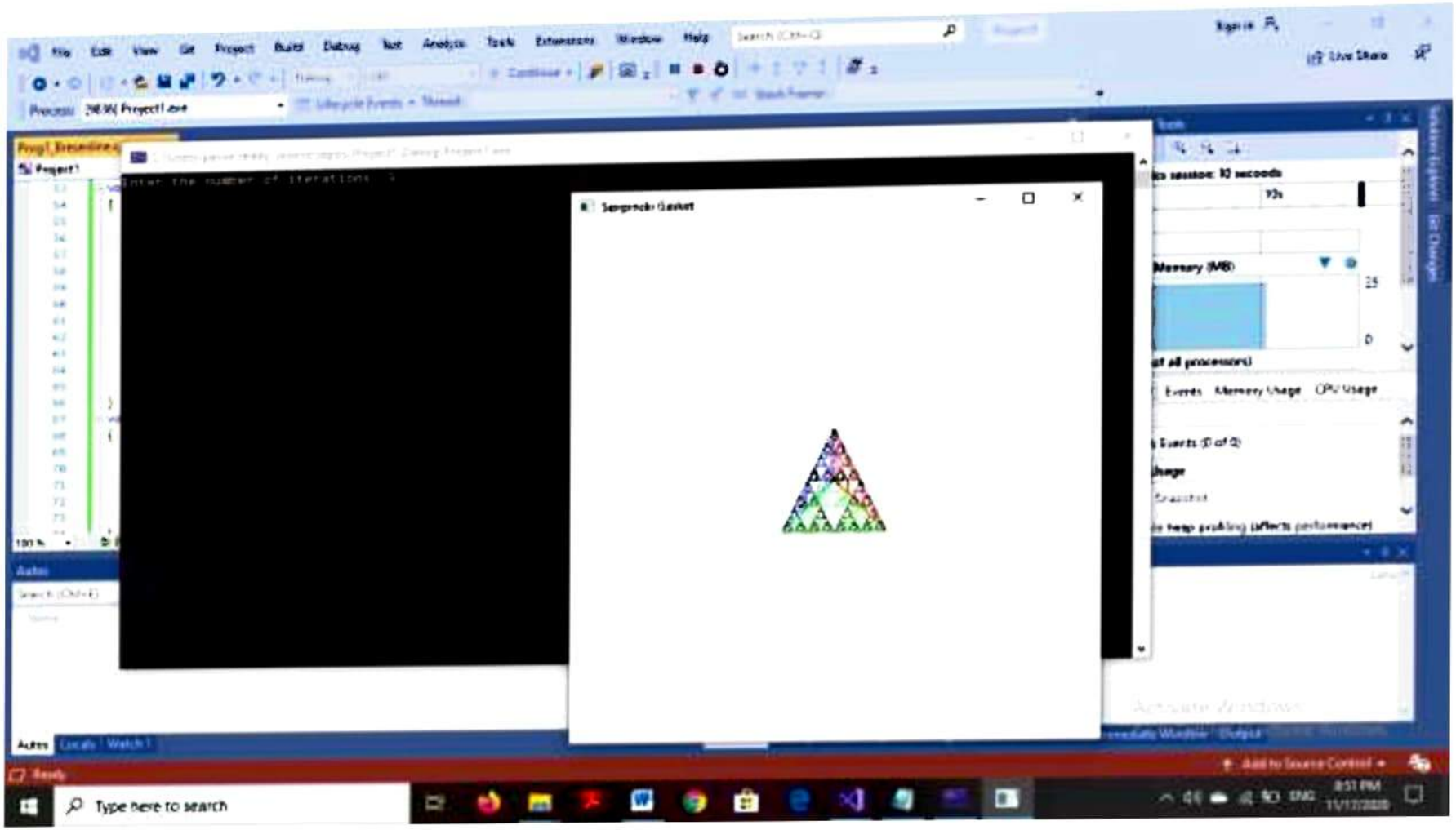
Teacher's Signature

```
}
```


Enter the number of iterations: 2
Sierpinski Gasket
Diagnostics session: 23 seconds
Process Memory (MB)
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Ready
Add to Source Control
Type here to search


Sierpinski Gasket
Diagnostic Tools
Type here to search


Sierpinski Gasket
Diagnostic Tools
Type here to search


Enter the number of iterations :
Sierpinski Gasket
Type here to search

Expt. No. 4 Date 2/11/20

Page No. 15

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 drawline(float x1, float y1, float x2, float y2)
{
    Sleep(100);
    glcolor3f(1,0,0);
    glBegin(GL_LINES);
    glVertex2f(x1, y1);
    glVertex2f(x2, y2);
    glEnd();
    glFlush();
}
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;
}
```

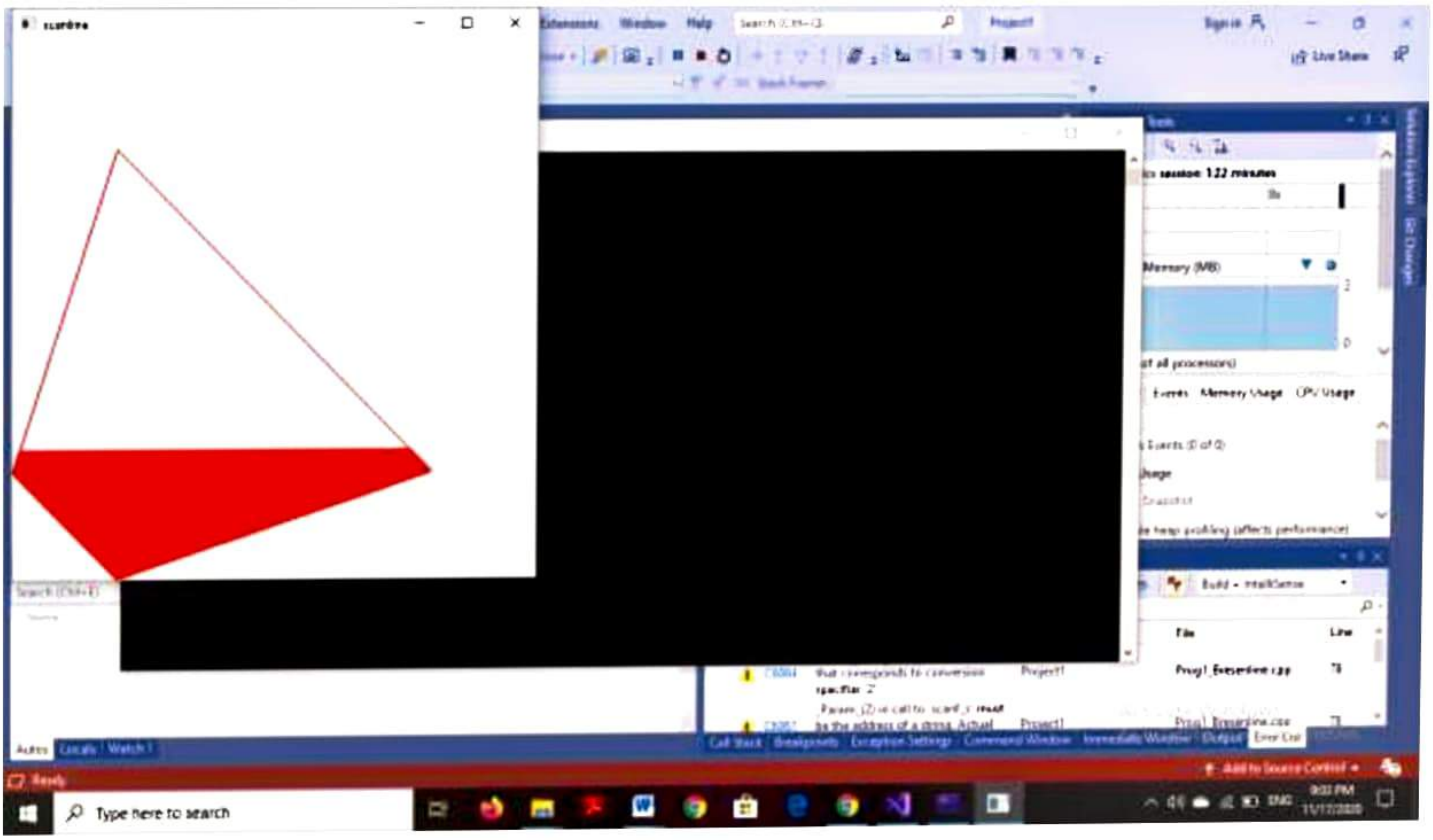
Teacher's Signature

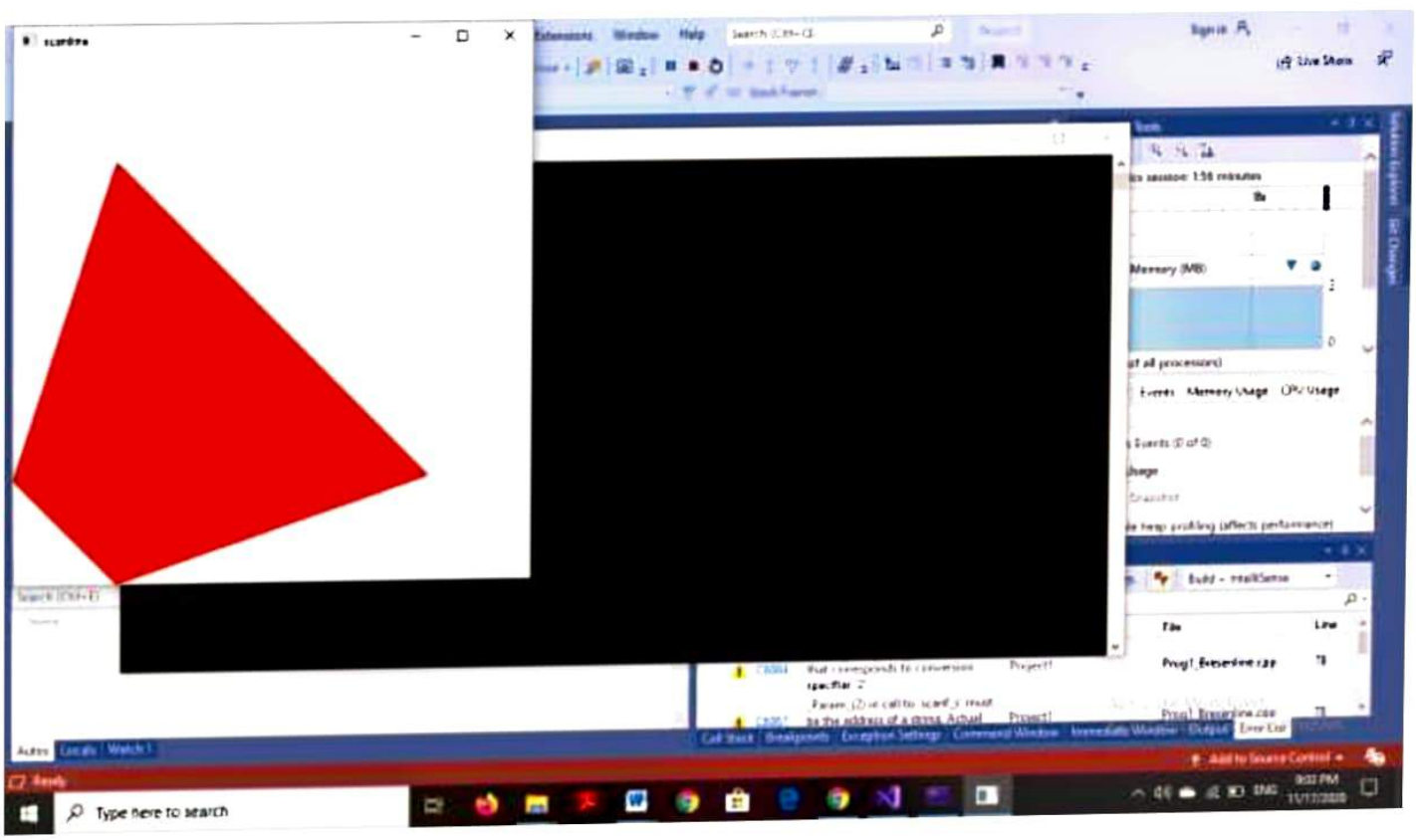
```
if(scanline > y1 && scanline < y2)
    int x[m++] = x1 + (scanline - y1) * (x2 - x1) /
                 (y2 - y1);
}

void scanfill(float x[], float y[]){
    for(int si = 0; si <= wy; si++){
        m = 0;
        for(int i = 0; i < n; i++){
            edgeDetect(x[i], y[i], x[(i+1) % n], y[(i+1) % n], si);
        }
        Sort(int x, (int x + m));
        if(m >= 2)
            for(int i = 0; i < m; i = i + 2)
                draw_line(int x[i], si, int x[i+1], si);
    }
}

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);
```

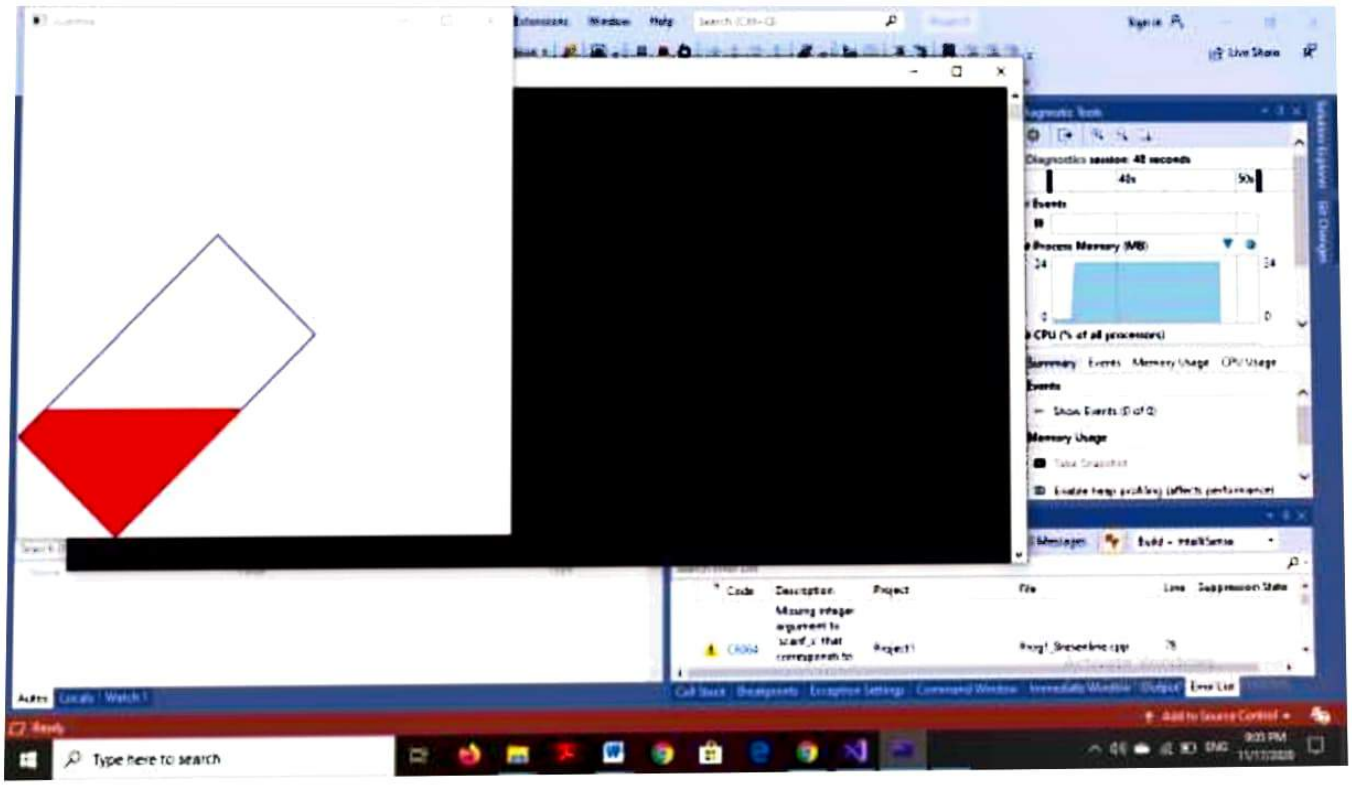
Teacher's Signature

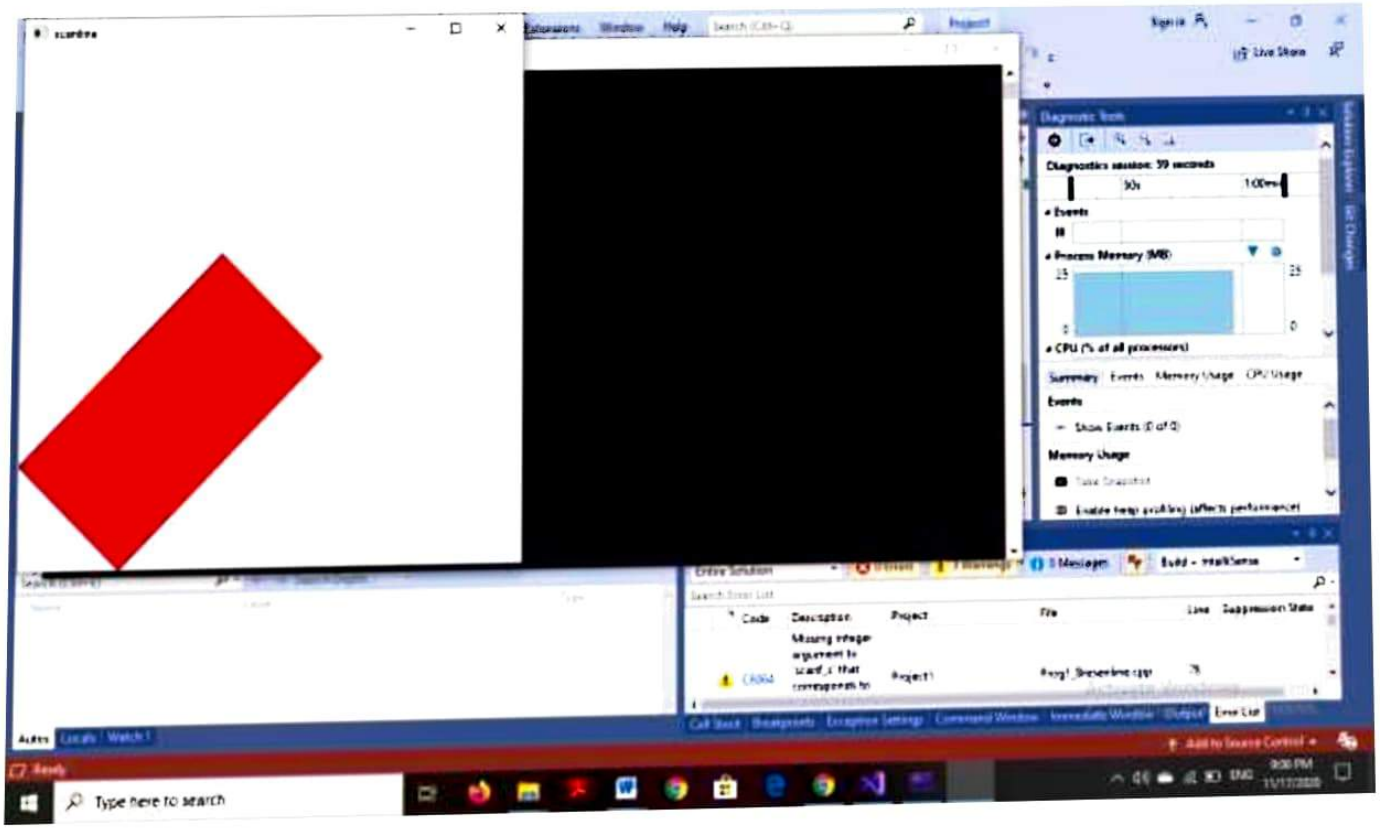
```
    glColor3f(0,0,1);
    glPointSize(1);
    gluOrtho2D(0, wx, 0, wy);
}

Void main(int ac, char* av[]) {
    glutInit(&ac, av);
    printf("Enter no of slides \n");
    scanf_s("%d", &n);
    printf("Enter Co-ordinates of endpoint : \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("Scanline");
    glutDisplayFunc(display_filled_polygon);
    myInit();
    glutMainLoop();
}
```

Output:






Type here to search


Type here to search

Date 9/11/20

Expt No. 5

Page No. 18

write a Pgm to to Create a house like figure & Perform the following operation

- Rotate it about a given fixed point using openGL transformation function
- Reflect it about an axis $y=mx+c$ using OPENGL transformation function.

```
#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);
    for(int i=0;i<11;i++)
        glVertex2fv(house[i]);
```

Teacher's Signature

```
glEnd();
glFlush();
glPushMatrix();
glTranslatef(100, 100, 0);
glRotatef(angle, 0, 0, 1);
glTranslatef(-100, -100, 0);
glColor3f(1, 1, 0);
glBegin(GL_LINE_LOOP);
for(int i=0; i<11; i++)
    glVertex2fv(house[i]);
glEnd();
glPopMatrix();
glFlush();
}

void display2()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_
        BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
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]);
```

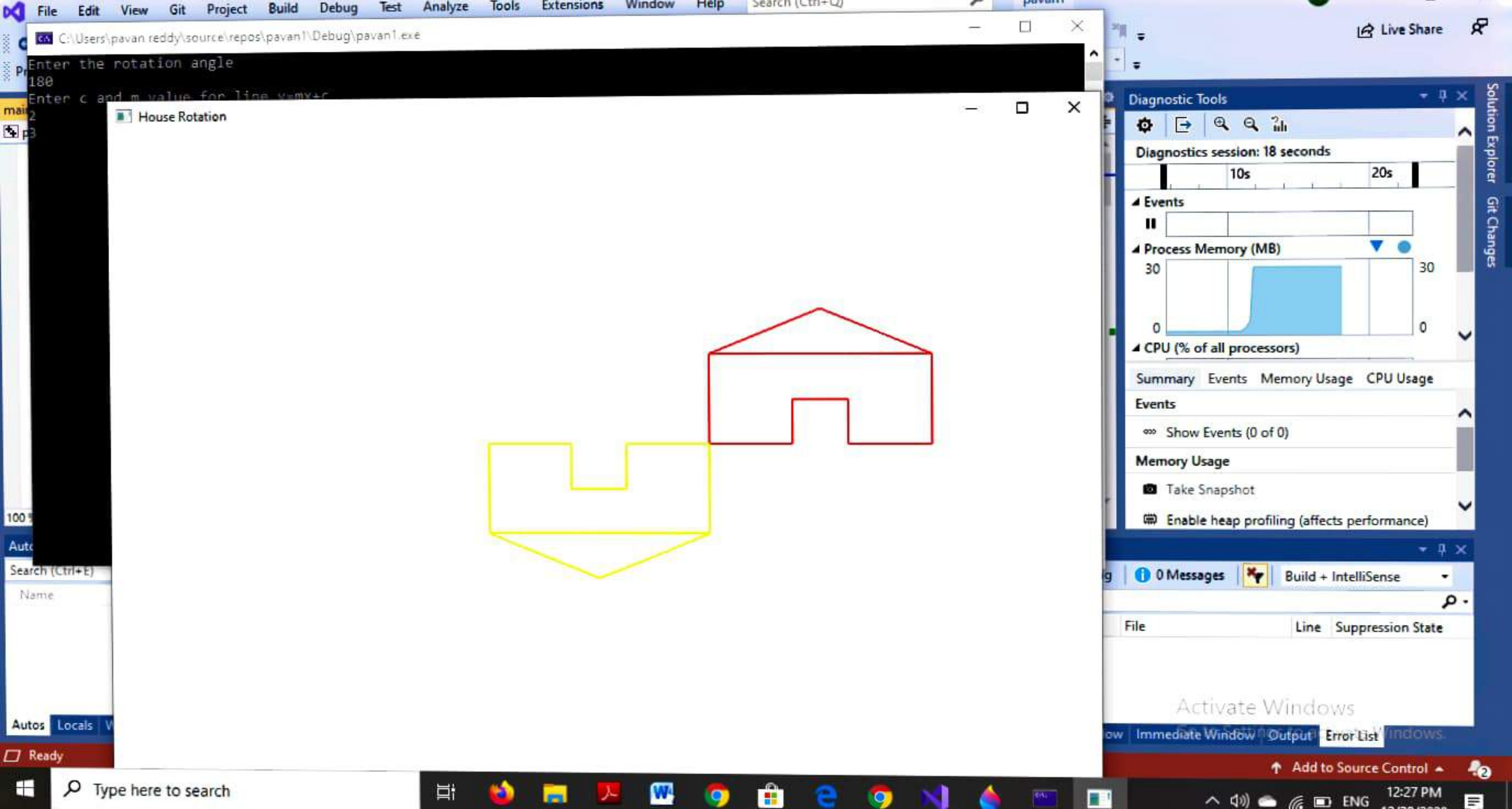
Teacher's Signature

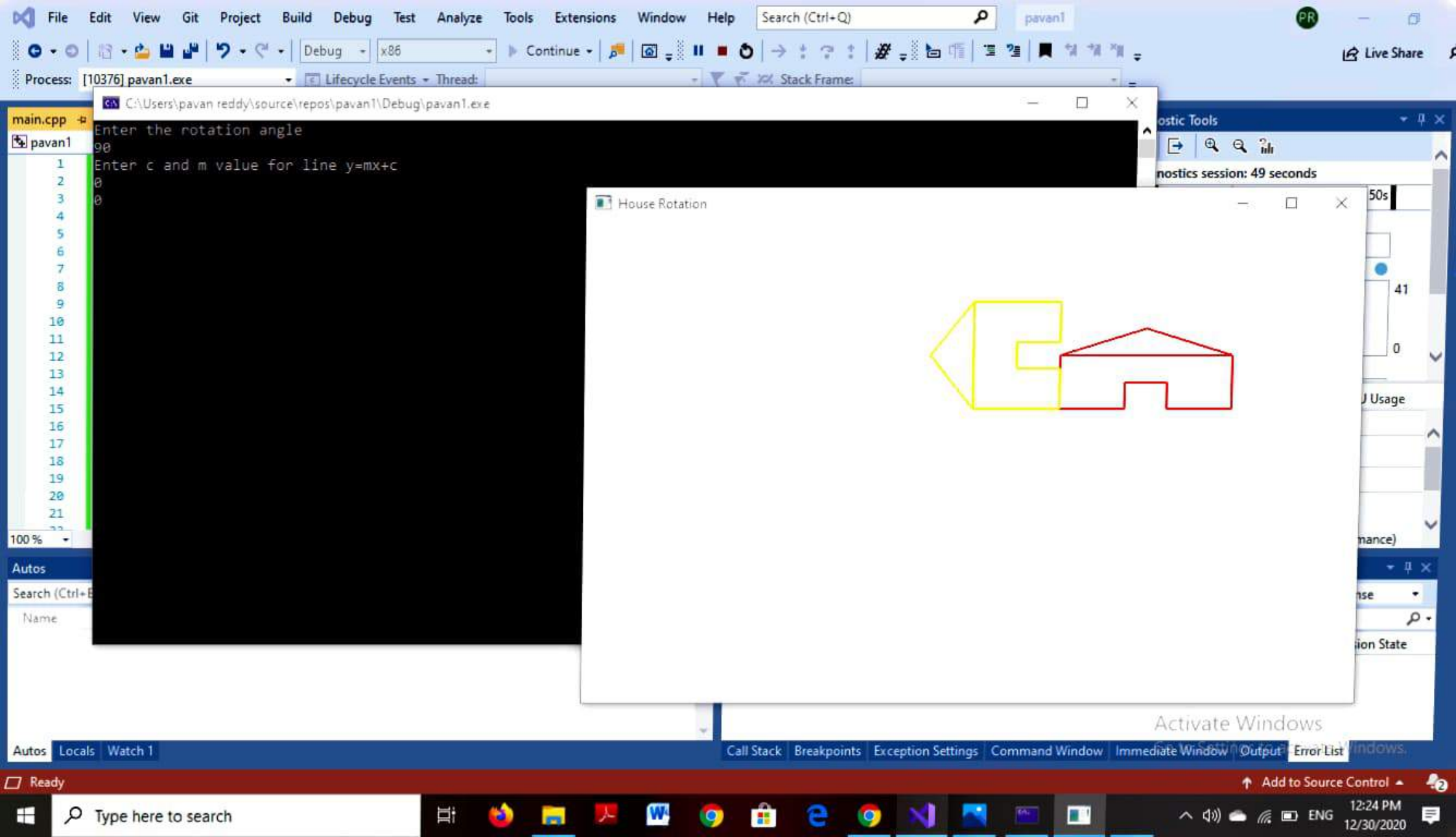
```
glEnd();
glFlush();
float x1 = 0, x2 = 500;
float y1 = m * x1 + c;
float y2 = m * x2 + c;
glColor3f(1, 1, 0);
glBegin(GL_LINES);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glEnd();
glFlush();
glPushMatrix();
glTranslatef(0, c, 0);
theta = atan(m);
theta = theta * 180 / 3.14;
glRotatef(theta, 0, 0, 1);
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();
}
void myinit() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);
```

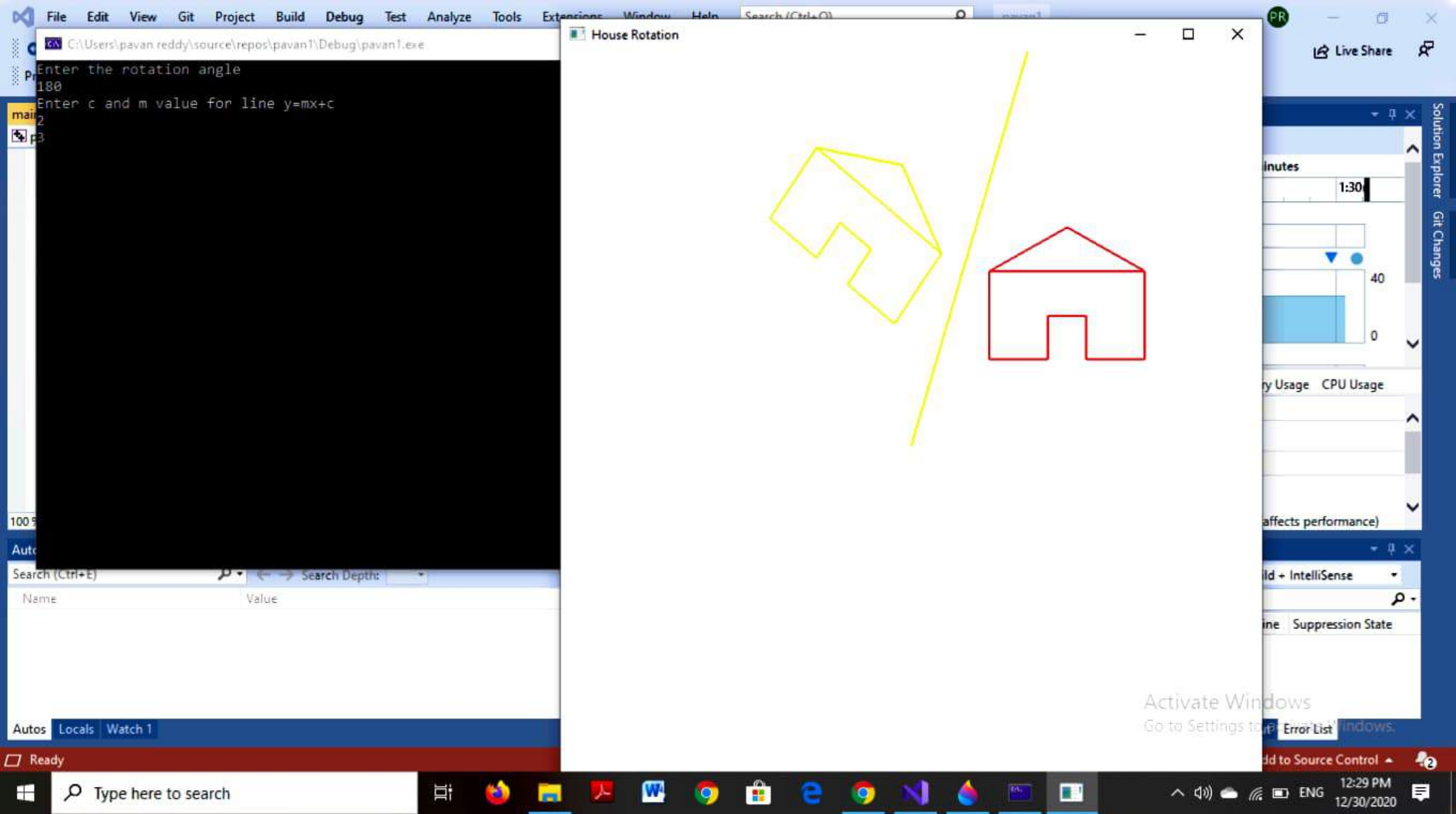
Teacher's Signature

```
    glLineWidth(2.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-450, 450, -450, 450);
}
void mouse(int btn, int state, int x, int y){
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_
        DOWN){
        display();
    }
    else if(btn==GLUT_RIGHT_BUTTON && state==
        GLUT_DOWN){
        display2();
    }
}
void main(int argc, char** argv)
{
    printf("Enter the rotation angle\n");
    scanf_s("%d", &angle);
    printf("Enter c & m value for line Y=mX+c\n");
    scanf_s("%f %f", &c, &m);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(900, 900);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("house Rotation");
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    myInit();
    glutMainLoop(); }
```

Teacher's Signature


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Live Share
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
Enter the rotation angle
180
Enter c and m value for line y=mx+c
2
3
House Rotation
Diagnostic Tools
Diagnostics session: 18 seconds
10s
20s
Events
Process Memory (MB)
30
0
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
0 Messages
Build + IntelliSense
File
Line
Suppression State
Immediate Window
Output
Error List
Add to Source Control
Ready
Autos
Locals
Type here to search
ENG
12:27 PM


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
pavan1
Debug
x86
Continue
Live Share
Process: [10376] pavan1.exe
Lifecycle Events
Thread:
Stack Frame:
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
Enter the rotation angle
90
Enter c and m value for line y=mx+c
0
0
House Rotation
main.cpp
pavan1
100 %
Autos
Search (Ctrl+E
Name
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Ready
Add to Source Control
Type here to search
ENG
12:24 PM
12/30/2020


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Live Share
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
Enter the rotation angle
180
Enter c and m value for line y=mx+c
2
3
House Rotation
Search (Ctrl+E)
Search Depth:
Name
Value
Autos
Locals
Watch 1
Ready
Error List
Suppression State
Solution Explorer
Git Changes
CPU Usage
Activate Windows
Type here to search
ENG
12:29 PM
12/30/2020

Expt. No. 6 Date 23/11/20

Page No. 22

write a pgm to implement the cohen-Sutherland 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 <stdlib.h>
#include <gl/glut.h>
#define outcode int
#define true 1
#define false 0
double xmin=50, ymin=50, xmax=100, ymax=100;
double xvmin=200, yvmin=200, xvmax=300, yvmax=300;
const int RIGHT=4;
const int LEFT=8;
const int TOP=1;
const int BOTTOM=2;
int n;
outcode ComputeOutCode (double x, double y)
{
    outcode code=0;
    if (y>ymax)
        code|=TOP;
    else if (y<ymin)
        code|=BOTTOM;
    if (x>xmax)
        code|=RIGHT;
    else if (x<xmin)
        code|=LEFT;
```

Teacher's Signature

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

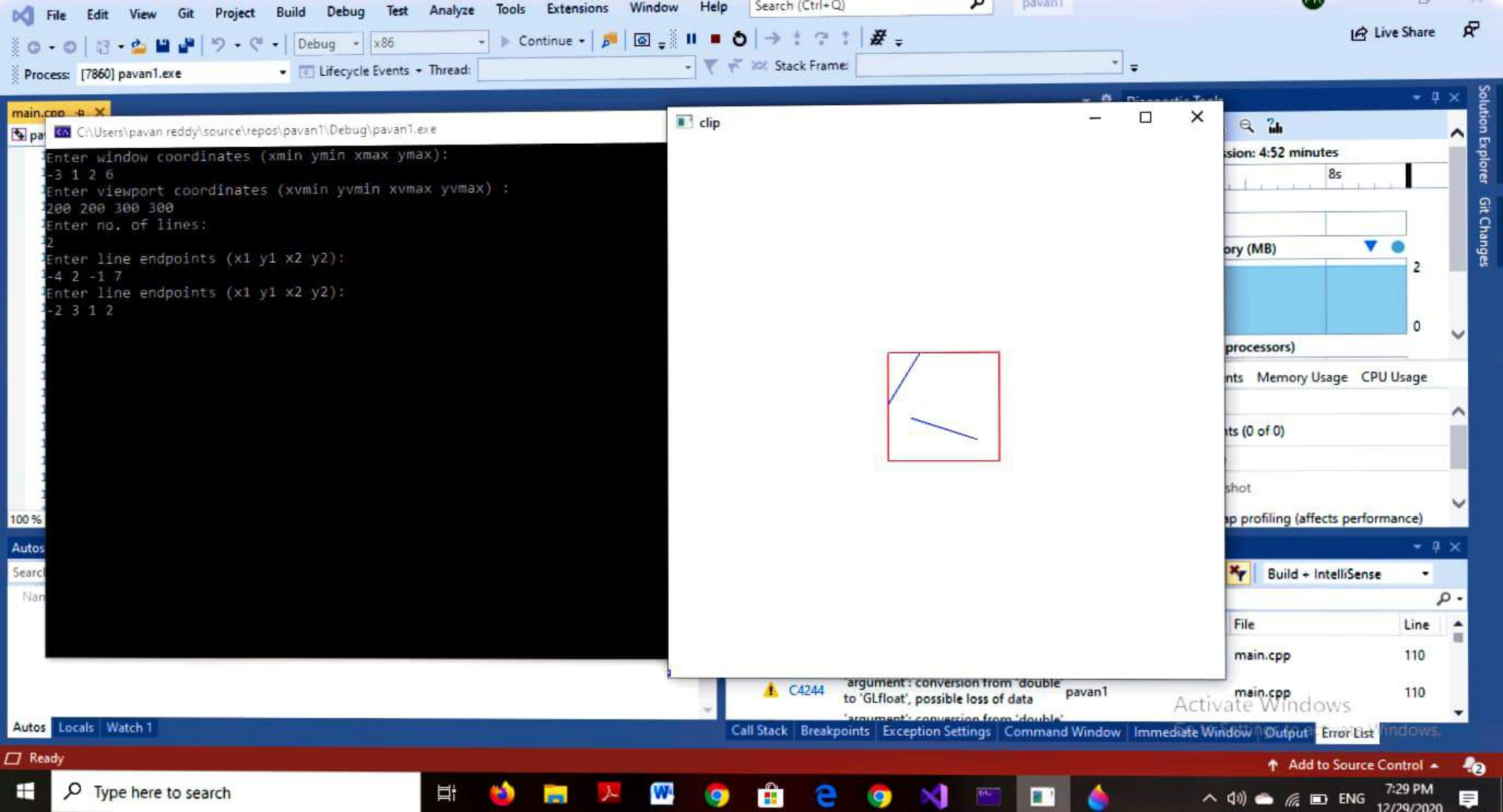
```
else if(outcodeout & RIGHT)
{
    y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
    x = xmax;
}
else {
    y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
    x = xmin;
}
if(outcodeout == outcode0)
{   x0 = x;
    y0 = y;
    outcode0 = computeoutcode(x0, y0);
else {
    x1 = x;
    y1 = y;
    outcode1 = computeoutcode(x1, y1);
}
}
} while(!done);
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 = xvmin + (y1 - ymin) * sy;
    glColor3f(1.0, 0.0, 0.0);
```

```
        glBegin(GL_LINE_LOOP);
        glVertex2f(xvmin, yvmin);
        glVertex2f(xvmax, yvmin);
        glVertex2f(xvmax, yvmax);
        glVertex2f(xvmin, yvmax);
        glEnd();
        glColour3f(0.0, 0.0, 1.0);
        glBegin(GL_LINES);
        glVertex2d(vx0, vy0);
        glVertex2d(vx1, vy1);
        glEnd();
    }
}
void display()
{
    double x0=60, y0=20, x1=80, y1=120;
    double x00=65, y00=65, x11=80, y11=80;
    double x02=25, y02=20, x12=30, y18=30;
    glClear(GL_COLOUR_BUFFER_BIT);
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2d(x0, y0);
    glVertex2d(x1, y1);
    glEnd();
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2f(xmin, ymin);
    glVertex2f(xmax, ymin);
    glVertex2f(xmax, ymax);
```

Teacher's Signature _____

```
        glVertex2f (xmin, ymax);
        glEnd();
        CohenSutherlandLineClipAndDraw (x0, y0, x1, y1);
        cohenSutherlandLineClipAndDraw (x00, y00, x11, y11);
        cohenSutherlandLineClipAndDraw (x02, y02, x10, y12);
        glFlush();
}
void myinit()
{
    glClearColor(1.0,1.0,1.0,1.0);
    glColor3f (1.0, 0.0, 0.0);
    glPointSize (1.0);
    glMatrixmode (GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D (0.0, 499.0, 0.0, 499.0);
}
void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (0, 0);
    glutCreateWindow ("Cohen Sutherland line clipping");
    glutDisplayFunc (display);
    myinit();
    glutMainLoop();
}
```

Teacher's Signature


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Debug
x86
Continue
Live Share
Process: [7860] pavan1.exe
Lifecycle Events
Thread:
Stack Frame:
main.cpp
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
Enter window coordinates (xmin ymin xmax ymax):
-3 1 2 6
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
200 200 300 300
Enter no. of lines:
2
Enter line endpoints (x1 y1 x2 y2):
-4 2 -1 7
Enter line endpoints (x1 y1 x2 y2):
-2 3 1 2
clip
Memory Usage
CPU Usage
Build + IntelliSense
File
Line
main.cpp
110
C4244
argument : conversion from 'double' to 'GLfloat', possible loss of data
pavan1
Activate Windows
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Ready
Add to Source Control
Type here to search
ENG
7:29 PM
12/29/2020

Date 29/11/20

Expt. No. 7

Page No. 27

write a pgm to implement the Liang-Barsky 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 <GL/glut.h>
double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;
int n;
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
```

Teacher's Signature

```
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);
```

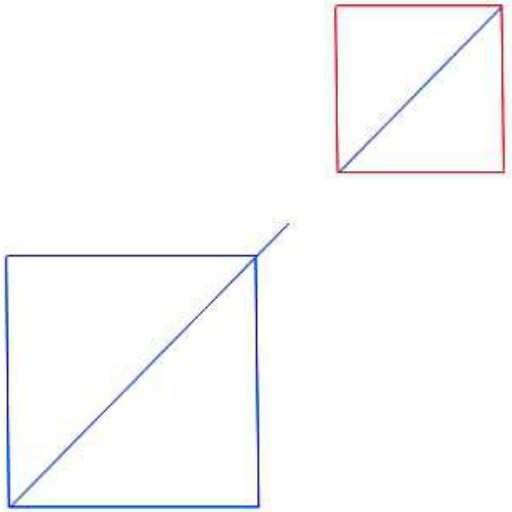
```
        glVertex2f (xmin, ymin);
        glVertex2f (xmax, ymin);
        glVertex2f (xmax, ymax);
        glVertex2f (xmin, ymax);
        glEnd();
        for(int i=0; i<n; i++)
            LiangBarskyLineClipAndDraw(ls[i].x1, ls[i].y1,
                                       ls[i].x2, ls[i].y2);
        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: (xmin, ymin
                xmax ymax)\n");
    scanf_s("%lf%lf%lf%lf", &xmin, &ymin,
        &xmax, &ymax);
```

Teacher's Signature _____

```
printf ("Enter viewport coordinates : (xvmin, yvmin,
                xvmax yvmax) \n");
scanf_s("%.1f %.1f %.1f", &xvmin, &yvmin, &xvmax,
                &yvmax);
printf ("Enter no of lines \n");
scanf_s("%d", &n);
for(int i=0; i<n; i++)
{
    printf ("Enter Co-ordinate : (x1, y1 x2 y2 \n");
    scanf_s("%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();
}
```

```
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
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:
2
Enter line endpoints (x1 y1 x2 y2):
100 100 250 250
Enter line endpoints (x1 y1 x2 y2):
200 200 270 270
```

clip



Date 7/12/20

Page No. 32

8

write a pgm 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_point
        [20][2];
const int MAX_POINTS = 20;
void drawpoly(int p[][2], int n) {
    glBegin(GL_POLYGON);
    for(int i=0; i<n; i++)
        glVertex2f(p[i][0], p[i][1]);
    glEnd();
}
int x_intersect(int x1, int y1, int x2, int y2, int x3,
        int y3, int x4, int y4)
{
    int num = (x1*y2 - y1*x2) * (x3 - x4) - (x1 - x2)
        * (x3*y4 - y3*x4);
    int den = (x1-x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
    return num / den;
}
int y_intersect(int x1, int y1, int x2, int y2, int x3,
        int y3, int x4, int y4);
{
    int num = (x1*y2 - y1*x2) * (y3 - y4) - (y1 - y2)
```

Teacher's Signature

```
        *(x3*y4 - y3*x4);
    int den = (x1-x2)*(y3-y4) - (y1-y2)*(x3-x4);
    return num / den;

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

```
        new_poly_size++;
        new_points[new_poly_size][0] = kx;
        new_points[new_poly_size][1] = ky;
        new_poly_size++;
    }
    else if (i_pos >= 0 && k_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
    {   // No points are added
    }
    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);
}
```

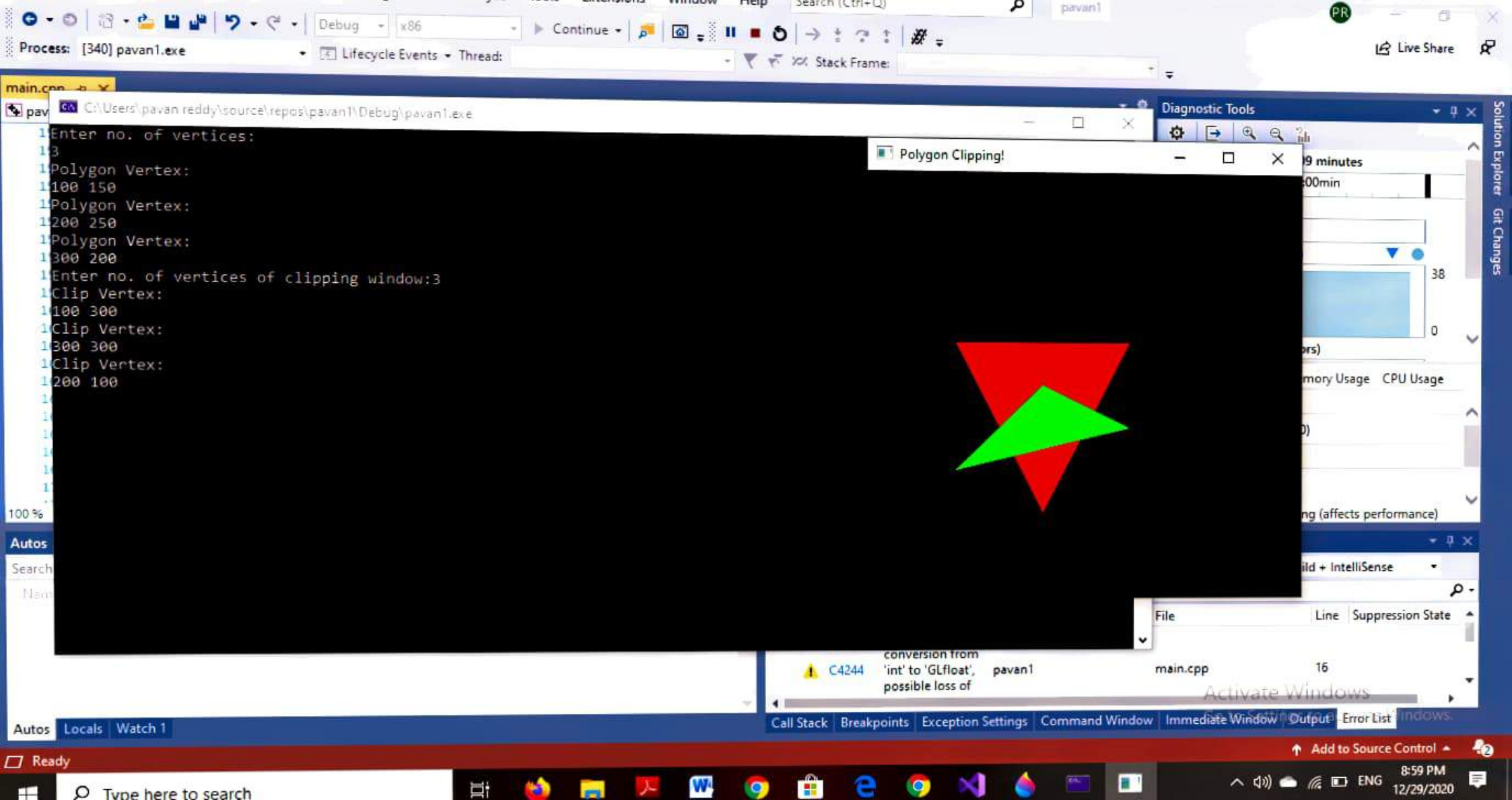
Teacher's Signature

```
void display()
{
    init();
    glColor3f(1.0f, 0.0f, 0.0f);
    drawPoly(clipper_points, clipper_size);
    glColor3f(0.0f, 1.0f, 0.0f);
    drawPoly(org_poly_points, org_poly_size);
    for(int i=0; i< clipperSize; i++)
    {
        int k= (i+1) % clipper_size;
        clip(polyx_points, poly_size, clipper_points[i][0],
             clipper_points[i][1], clipper_points[k][0],
             clipper_points[k][1]);
    }
    glColor3f(0.0f, 0.0f, 1.0f);
    drawPoly(poly_points, poly_size);
    glFlush();
}
int main(int argc, char* argv[])
{
    printf("Enter no of vertices: \n");
    scanf_s("%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];
```

Teacher's Signature _____

```
    org_poly_points[i][i] = poly_points[i][i];
}
printf("Enter no of vertices of clipping window");
scanf_s("%d", &clipper_size);
for(int i=0; i<clipper_size; i++)
{
    Printf("clip vertex : \n");
    scanf_s("%d%d", &clipper_points[i][0], &clipper
            _points[i][i]);
}
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400, 400);
glutInitWindowPosition(100, 100);
glutCreateWindow("polygon clipping");
glutDisplayFUNC(display);
glutMainLoop();
return 0;
}
```

Teacher's Signature _____


C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
Enter no. of vertices:
3
Polygon Vertex:
100 150
Polygon Vertex:
200 250
Polygon Vertex:
300 200
Enter no. of vertices of clipping window:3
Clip Vertex:
100 300
Clip Vertex:
300 300
Clip Vertex:
200 100
Polygon Clipping!
Diagnostic Tools
C4244
conversion from 'int' to 'GLfloat', possible loss of
pavan1
main.cpp
16
Ready

Date 14/12/20

Expt. No. 9

Page No. 37

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

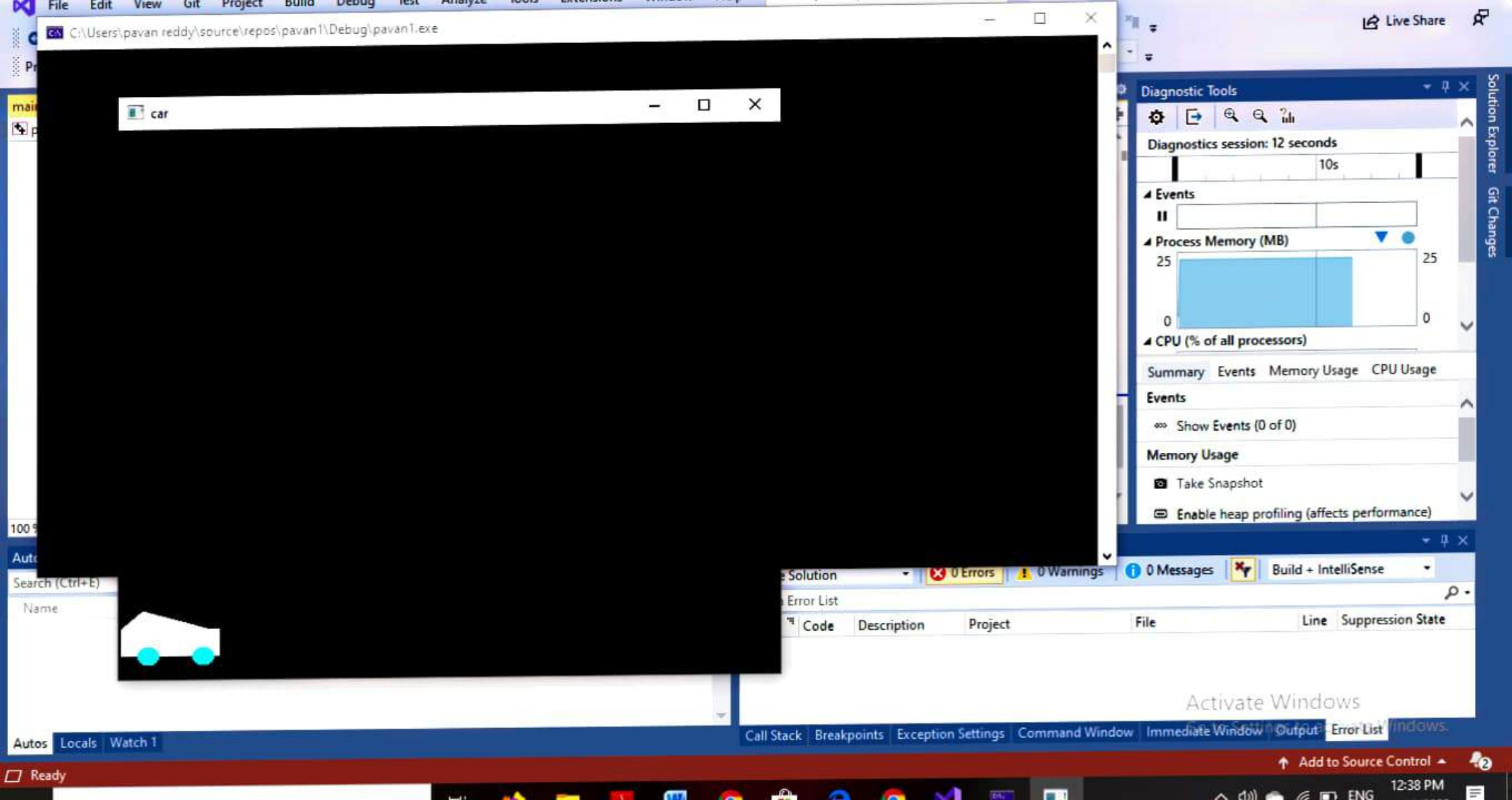
```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
#define CAR 1
#define WHEEL 2
float S=1;
void carlist() {
        glNewList(CAR, GL_COMPILE);
        glColor3f(1,1,1);
        glBegin(GL_POLYGON);
        glVertex3f(0,25,0);
        glVertex3f(90,25,0);
        glVertex3f(90,55,0);
   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

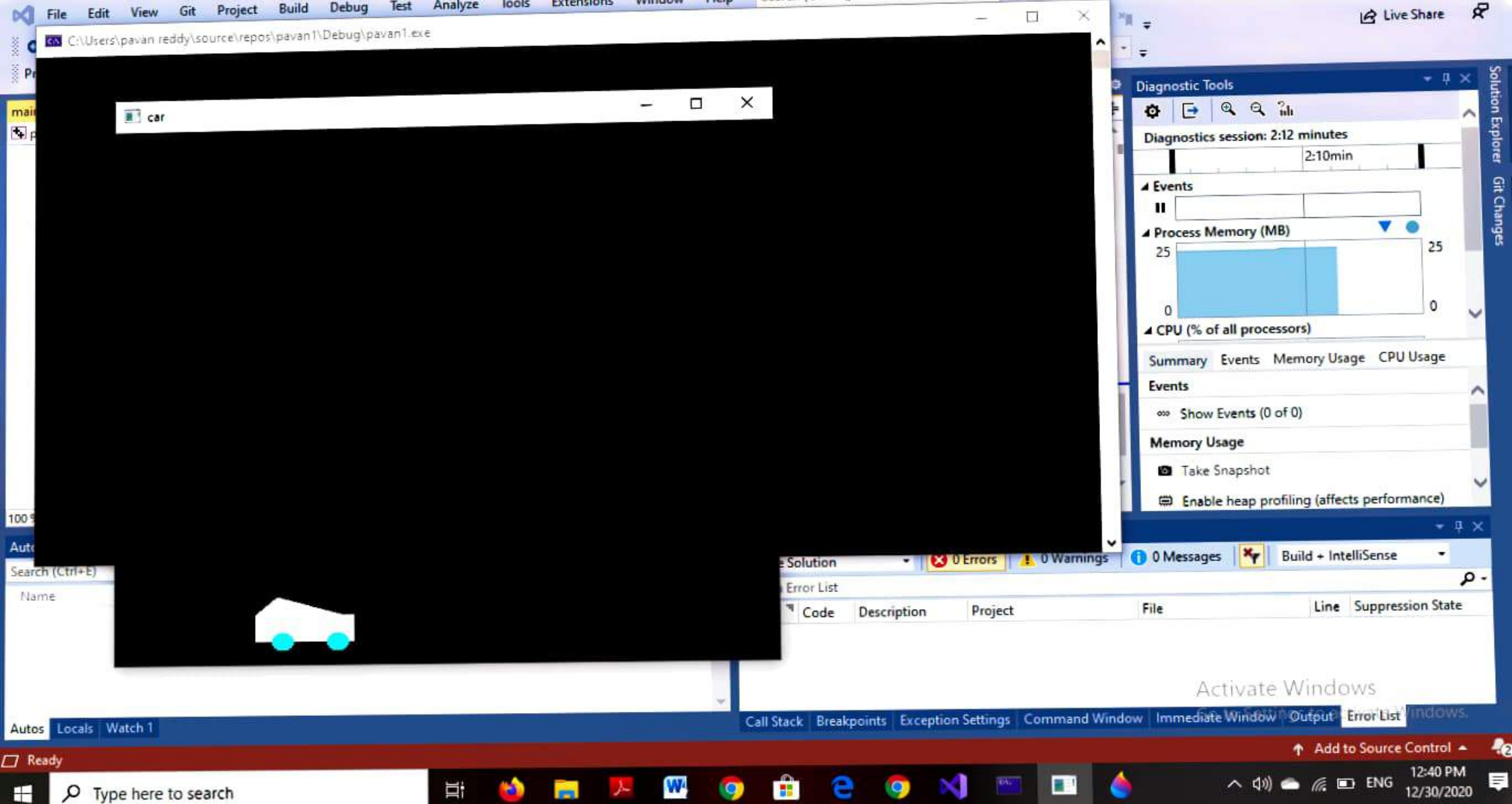
Teacher's Signature

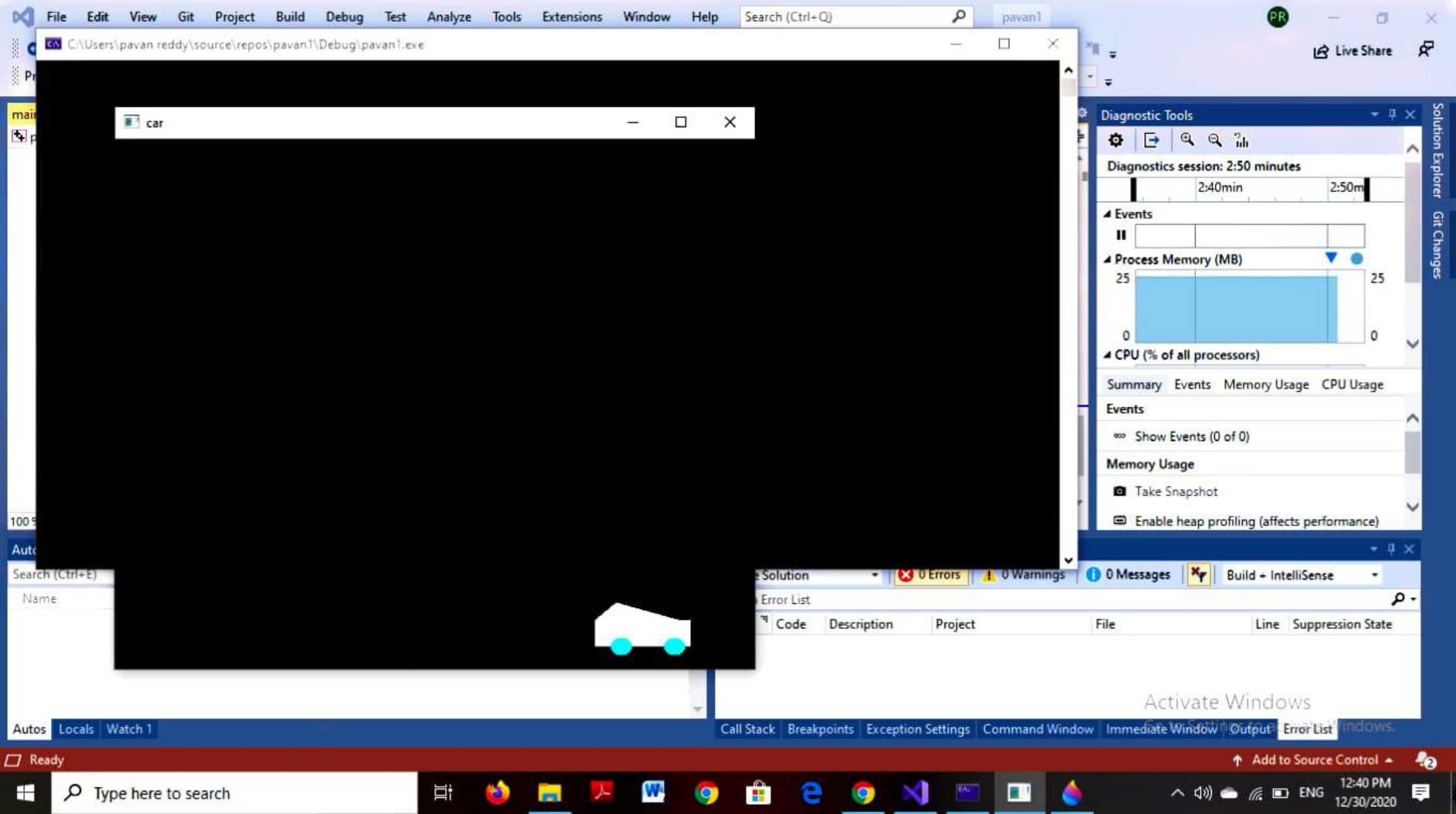
```
void mykeyboard (unsigned char key, int x, int y) {
        Switch (key) {
            case 't': glutPostRedisplay();
                      break;
            case 'q' : exit(0);
            default : break;
        }
}
void myinit() {
        glClearColor(0, 0, 0, 0);
        glOrtho (0, 600, 0, 600, 0, 600);
}
void draw_wheel() {
        glColor3f (0, 1, 1);
        glutSolidSphere (10, 25, 25);
}
void moveCar(float s) {
        glTranslatef (s, 0.0, 0.0);
        glCallList (CAR);
        glPushMatrix();
        glTranslatef (25, 25, 0.0);
        glCallList (WHEEL);
        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){
        s+=5;
        myDisp();
    }
    else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN){
        s+=2;
        myDisp();
    }
}
int 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);
    glutMainLoop();
}
```

Teacher's Signature _____


C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
car
Diagnostic Tools
Diagnostics session: 12 seconds
10s
Events
Process Memory (MB)
25
0
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
Live Share
Solution Explorer
Git Changes
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Error List
Code
Description
Project
File
Line
Suppression State
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Autos
Locals
Watch 1
Search (Ctrl+E)
Name
Activate Windows
Ready
Add to Source Control
12:38 PM


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Live Share
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
car
Diagnostic Tools
Diagnostics session: 2:12 minutes
2:10min
Events
Process Memory (MB)
25
0
CPU (% of all processors)
Summary Events Memory Usage CPU Usage
Events
Show Events (0 of 0)
Memory Usage
Take Snapshot
Enable heap profiling (affects performance)
Solution Explorer
Git Changes
Search (Ctrl+E)
Name
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Error List
Code
Description
Project
File
Line
Suppression State
Activate Windows
Autos Locals Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List
Ready
Add to Source Control
Type here to search
ENG
12:40 PM
12/30/2020


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
pavan1
Live Share
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
car
Diagnostic Tools
Diagnostics session: 2:50 minutes
2:40min
2:50m
Events
Process Memory (MB)
25
0
CPU (% of all processors)
Summary
Events
Memory Usage
CPU Usage
Show Events (0 of 0)
Take Snapshot
Enable heap profiling (affects performance)
Solution Explorer
Git Changes
Search (Ctrl+E)
Name
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Error List
Code
Description
Project
File
Line
Suppression State
Activate Windows
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Ready
Add to Source Control
Type here to search
ENG
12:40 PM
12/30/2020

Date 21/12/20

Expt. No. 10 Page No. 40

write a pgm to create a color cube & spin it using openGL transformation.

```
#include<stdlib.h>
#include<GL/glut.h>
#include<gl/GL.h>
#include<gl/GLU.h>
#include<time.h>
GLfloat vertices = {-1.0,-1.0,-1.0,1.0,-1.0,-1.0,1.0,1.0,
                    -1.0,-1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,
                    -1.0,1.0,1.0,1.0,1.0,-1.0,1.0,1.0};
GLfloat normals[] = {-1.0,-1.0,-1.0,1.0,-1.0,-1.0,1.0,1.0,
                     -1.0,-1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,
                     -1.0,1.0,1.0,1.0,1.0,-1.0,1.0,1.0};
GLfloat colors[] = {0.0,0.0,0.0,1.0,0.0,0.0;1.0,1.0,0.0,
                    0.0,1.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,
                    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)
{
```

Teacher's Signature

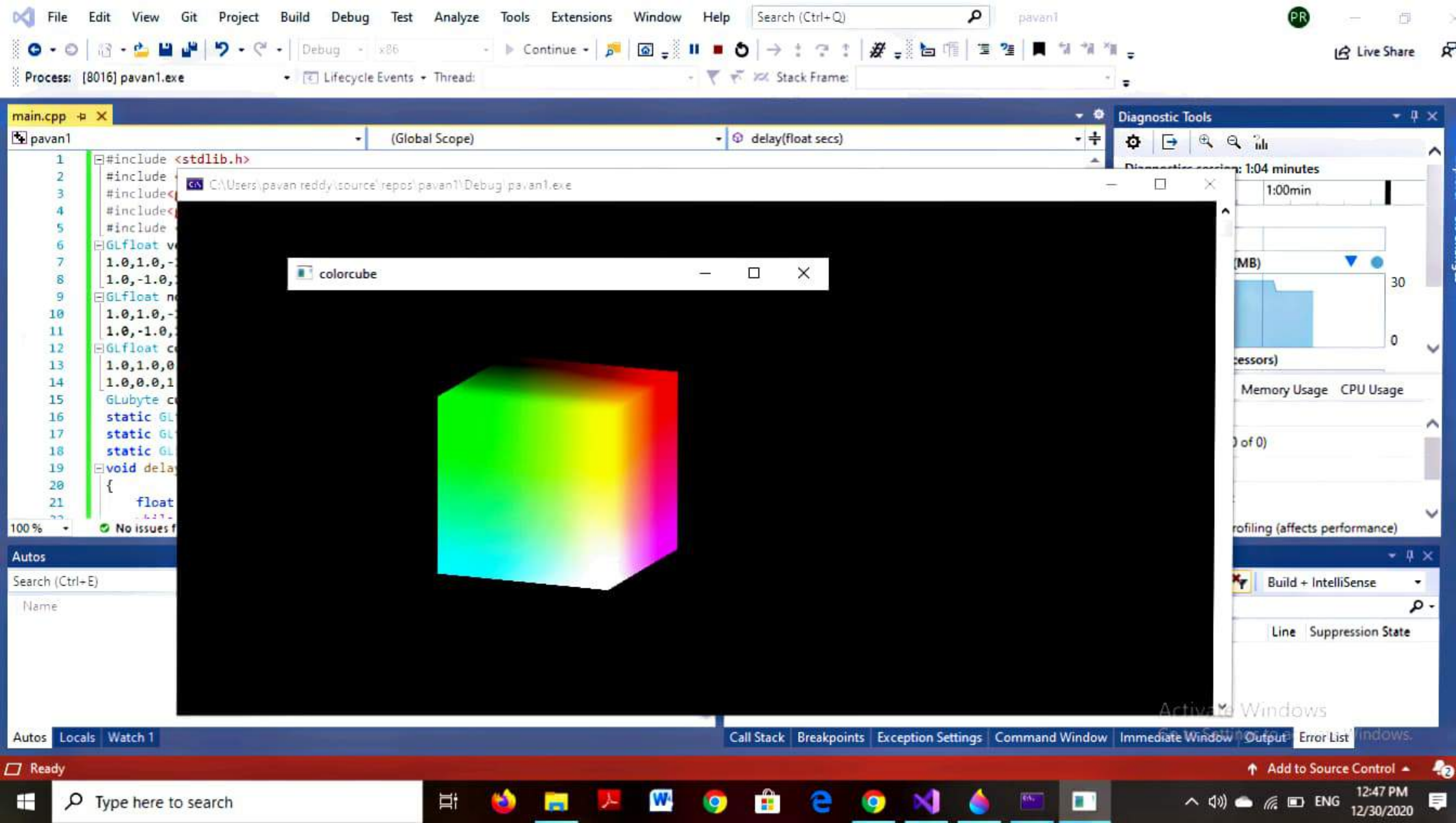
```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_
        BIT);
glLoadIdentity();
glRotatef(theta[0], 1.0, 0.0, 0.0);
glRotatef(theta[1], 0.0, 1.0, 0.0);
glRotatef(theta[2], 0.0, 0.0, 1.0);
glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE,
               cubeIndices);
glBegin(GL_LINES);
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(1.0, 1.0, 1.0);
glEnd();
glFlush();
}

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

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

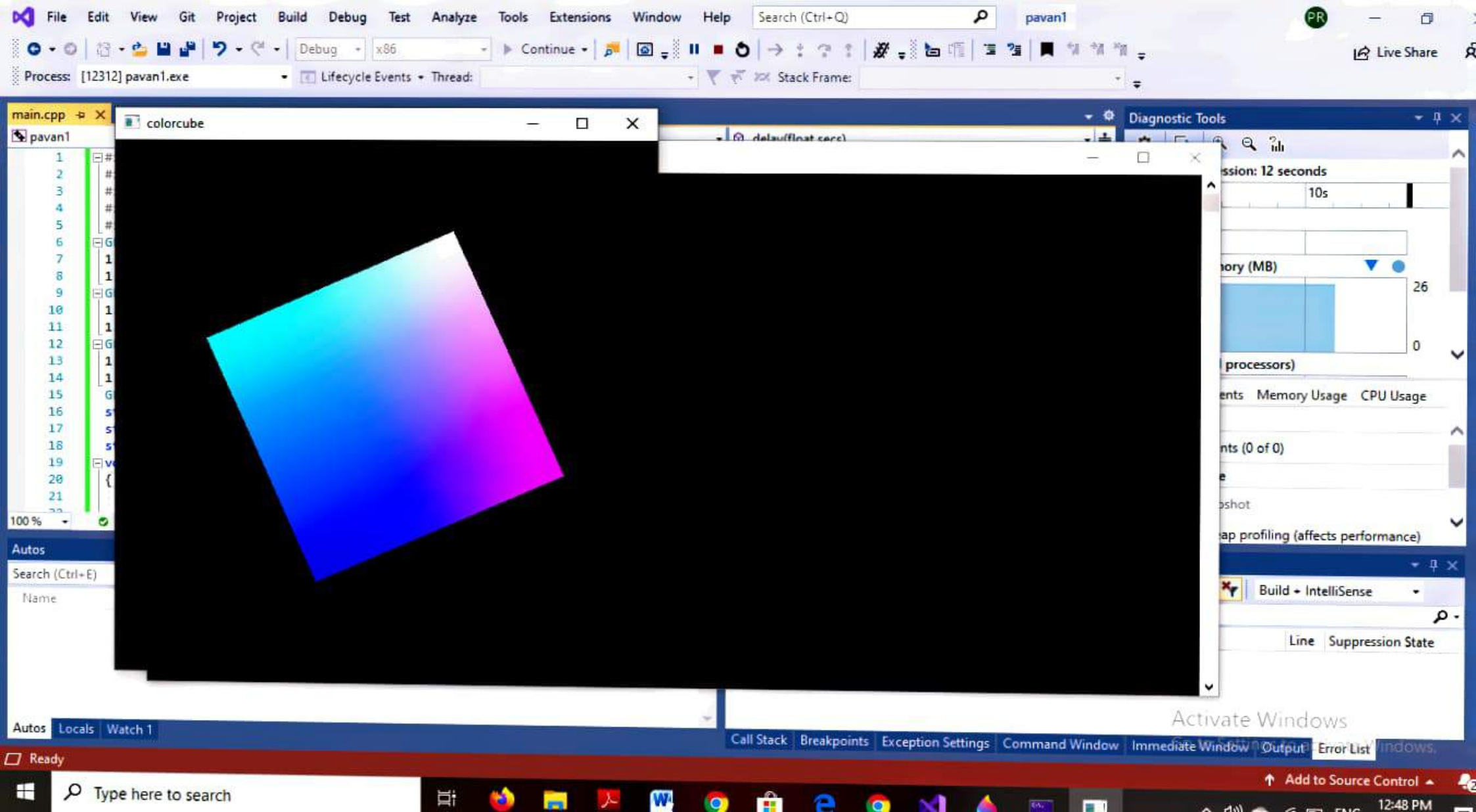
Teacher's Signature

```
void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity()
    if (w<=h)
        glOrtho(-2.0, 2.0, -2.0 * (GLfloat)h / (GLfloat)w,
                2.0 * (GLfloat)h / (GLfloat)w, -10.0,
                10.0);
    else
        glOrtho(-2.0 * (GLfloat)w / (GLfloat)h, 2.0 * (GLfloat)
                w / (GLfloat)h, -2.0, 2.0, -10.0, 10.0);
    glMatrixMode (GL_MODELVIEW);
}
void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (100, 100);
    glutInitWindowSize (500, 500);
    glutCreateWindow ("ColorCube");
    glutReshapeFunc (myReshape);
    glutDisplayFunc (displaySingle);
    glutIdleFunc (spinCube);
    glutMouseFunc (mouse);
    glEnable (GL_DEPTH_TEST);
    glEnableClientState (GL_COLOR_ARRAY);
    glEnableClientState (GL_NORMAL_ARRAY);
    glEnableClientState (GL_VERTEX_ARRAY);
```

Teacher's Signature

```
glVertexPointer(3, GL_FLOAT, 0, Vertices);
glColorPointer(3, GL_FLOAT, 0, Colors);
glNormalPointer(3, GL_FLOAT, 0, Normals);
glColor3f (1.0, 1.0, 1.0);
glutMainLoop();
}
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Live Share
Process: [8016] pavan1.exe
main.cpp
pavan1
(Global Scope)
delay(float secs)
#include <stdlib.h>
C:\Users\pavan reddy\source\repos\pavan1\Debug\pavan1.exe
colorcube
Diagnostic Tools
Memory Usage
CPU Usage
Autos
Search (Ctrl+E)
Name
Build + IntelliSense
Line
Suppression State
Autos
Locals
Watch 1
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Ready
Add to Source Control
Type here to search
ENG
12:47 PM
12/30/2020


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
pavan1
Debug
x86
Continue
Live Share
Process: [12312] pavan1.exe
Lifecycle Events
Thread:
Stack Frame:
main.cpp
pavan1
colorcube
Diagnostic Tools
10s
26
0
processors)
Memory Usage
CPU Usage
Autos
Search (Ctrl+E)
Name
Autos
Locals
Watch 1
Build + IntelliSense
Line
Suppression State
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List
Activate Windows
Ready
Add to Source Control
Type here to search
12:48 PM

Date 28/12/20

Prgm. No. 11

Page No. 44

Create a menu with 3 entries named curves, Color & Quit. The entry curves has a sub menu which has 4 entries namely Limacon, cardiod, Three-Leaf, & spiral. The color menu has sub menu with all eight colors of RGB Color model. write a Pgm to Create the above hierarchical menu & attach appropriate services to each menu entries with mouse button.

```
#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);
        glOrtho2D (0.0, 200.0, 0.0, 150.0);
}
void lineSegment(SScreenPt P1, SScreenPt P2) {
    glBegin(GL_LINES);
    glVertex2i (P1.X, P1.Y);
    glVertex2i (P2.X, P2.Y);
```

Teacher's Signature

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

Teacher's Signature

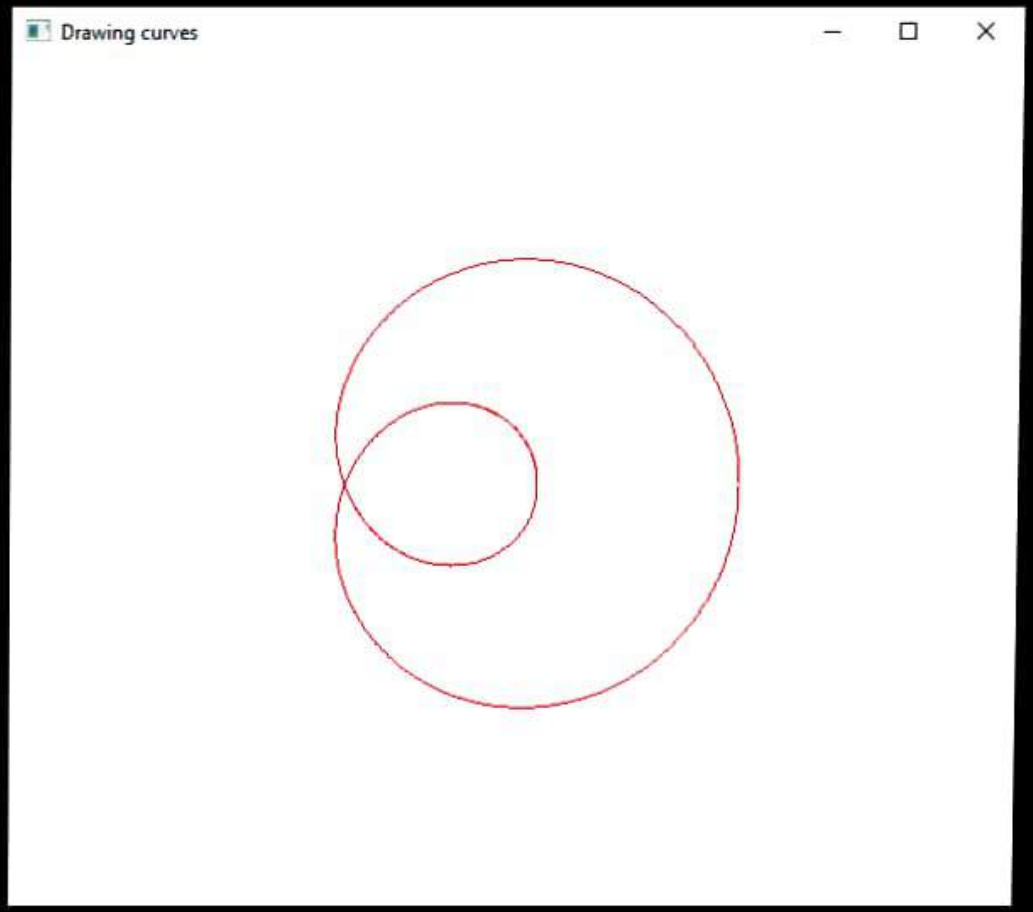
```
        default : break;
    }
    curvept[i].x = x0+r*cos(theta);
    curvept[i].y = y0+r*sin(theta);
    LineSegment(curvept[0], curvept[i]);
    &
    curvept[0].x = curvept[i].x;
    curvept[0].y = curvept[i].y;
    theta += dtheta;
  }
}
void colorMenu(int id) {
    Switch(id) {
     case 0 : break;
     case 1 :
            red=0;
            green=0; blue=1;
            break;
    case 2 :
            red=0;
            green=1;
            blue=0; break;
   case 4 : red=1;
            green=0;
            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(){ }
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);
glutInitWindowPosition(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);
int colorId = glutCreateMenu(colorMenu);
  glutAddMenuEntry("Red",4);
  glutAddMenuEntry("Green",2);
  glutAddMenuEntry("Blue",1);
  glutAddMenuEntry("Yellow",6);
  glutAddMenuEntry("Black",0);
  glutAddMenuEntry("Cyan",3);
  glutAddMenuEntry("Magenta",5);
  glutAddMenuEntry("White",7);
  glutAttachMenu(GLUT_LEFT_BUTTON);
  glutCreateMenu(main_menu);
  glutAddSubMenu("drawCurve", curveId);
  glutAddSubMenu("Colors", colorId);
  glutAddMenuEntry("quit",3);
  glutAttachMenu(GLUT_LEFT_BUTTON);
  myinit();
  glutDisplayFunc(mydisplay);
  glutReshapeFunc(myreshape);
  glutMainLoop();
}
```

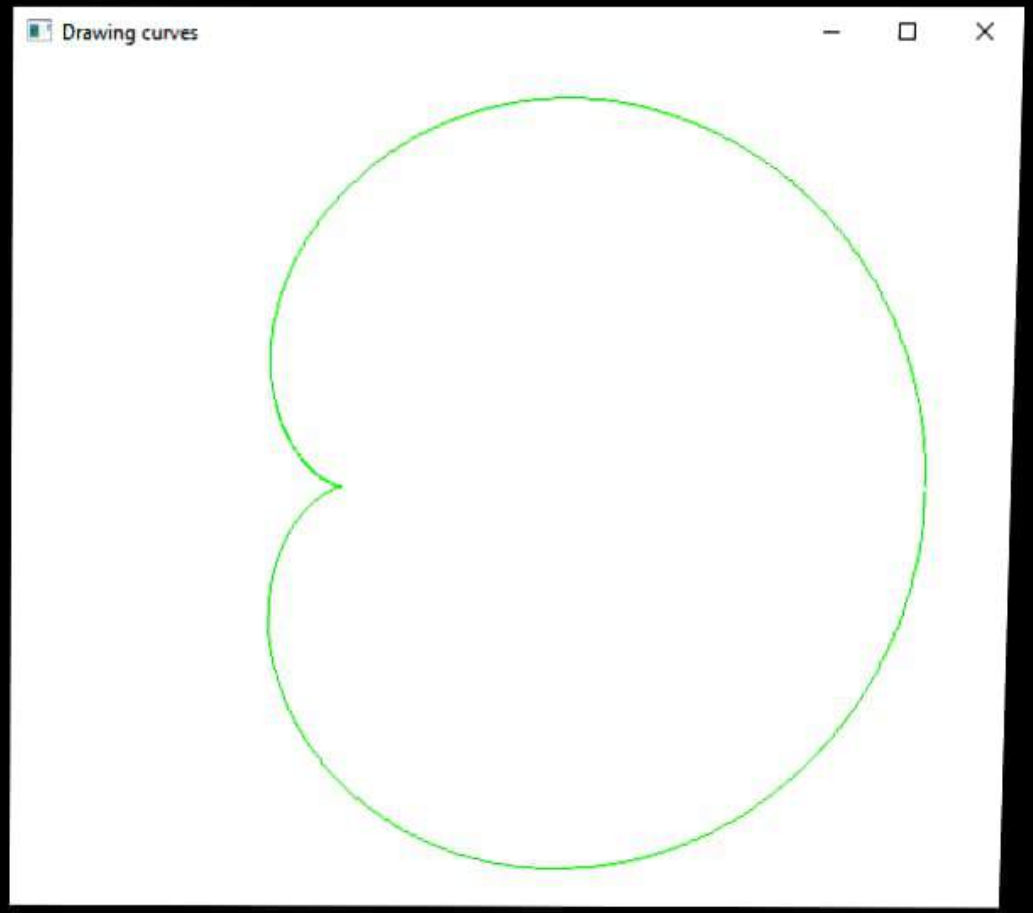
Teacher's Signature

C:\Users\Vaibhav\source\repos\labpgm11_CurveMenu\Debug\labpgm11_CurveMenu.exe

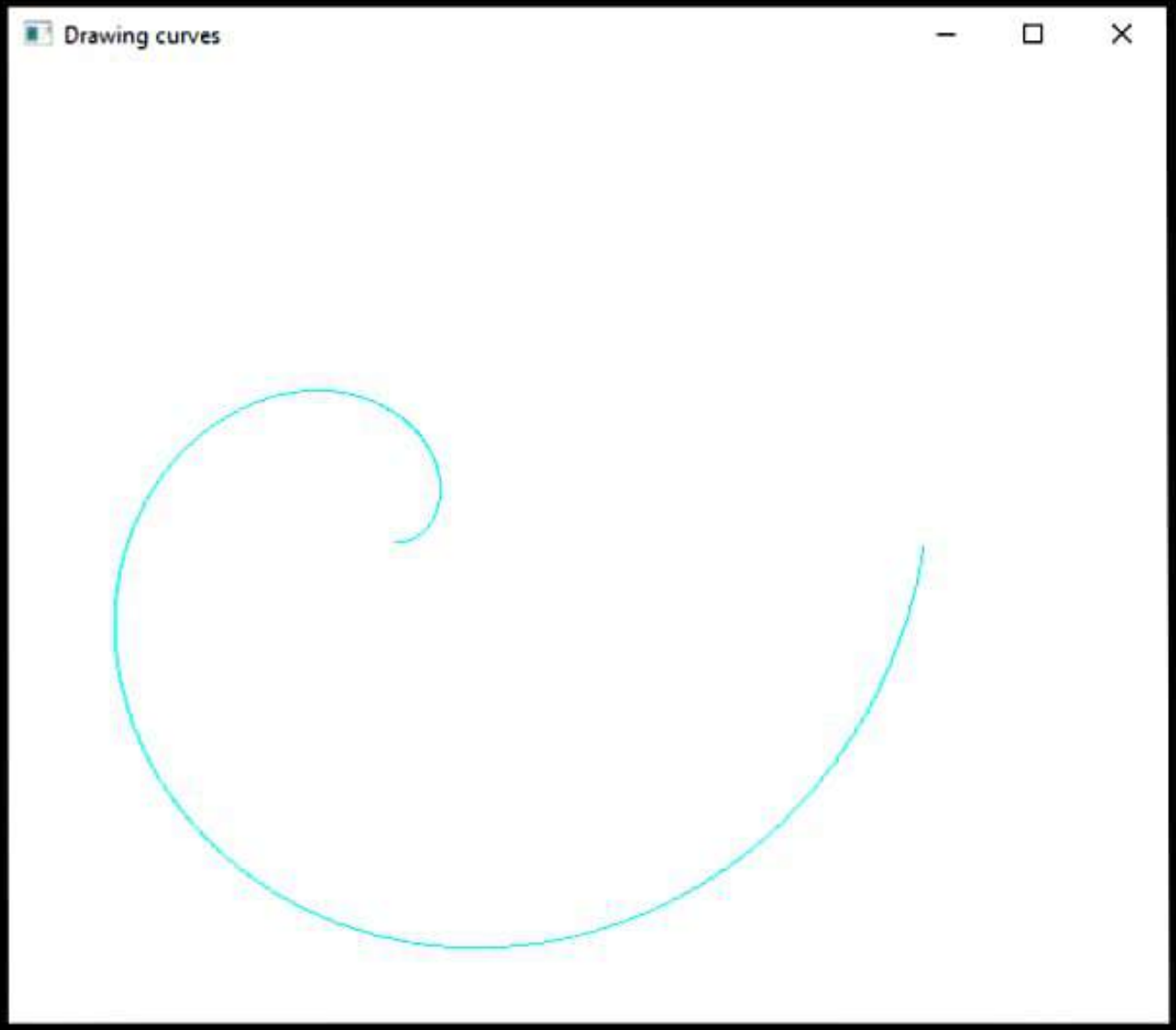


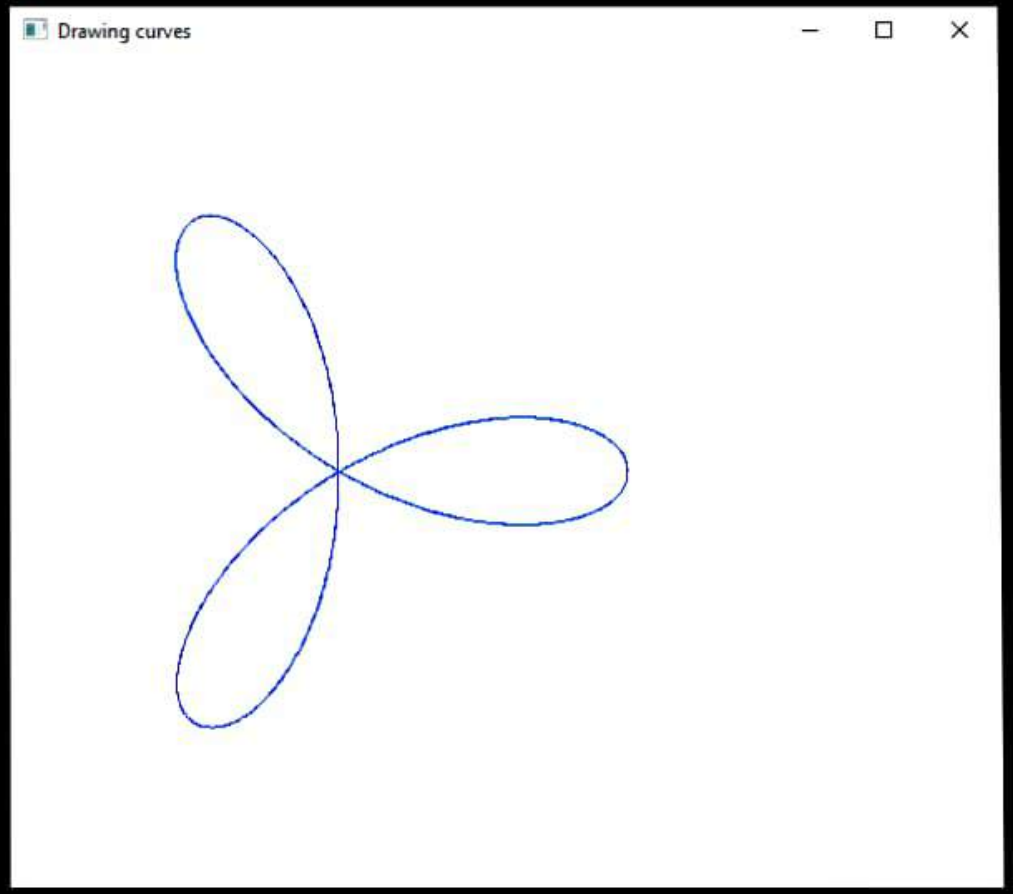
Type here to search

ENG
14:52
05-12-2020



C:\Users\Vaibav\source\repos\labpgm11_CurveMenu\Debug\labpgm11_CurveMenu.exe




Drawing curves

12

Date 30/12/20

Page No. 49

write a pgm to construct Bezier curve control points are supplied through Keyboard / Mouse.

```
#include <iostream>
#include <math.h>
#include <gl/glut.h>
using namespace std;
float x, y, n, X1[4], YC[4];
int flag = 0;
void myInit() {
    glClearColor(1,1,1,1);
    glColor3f(1,1,1);
    glPointSize(5);
    gluOrtho2D(0,500,0,500);
}
void drawPixel(float x, float y) {
    glBegin(GL_POINTS);
    glVertex2f(x,y);
    glEnd();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    int i;
    double t;
    glColor3f(0,0,0);
    glBegin(GL_POINTS);
    for(t=0; t<1; t=t+0.005) {
        double xt = pow(1-t,3) * X1[0] + 3*t*pow(
```

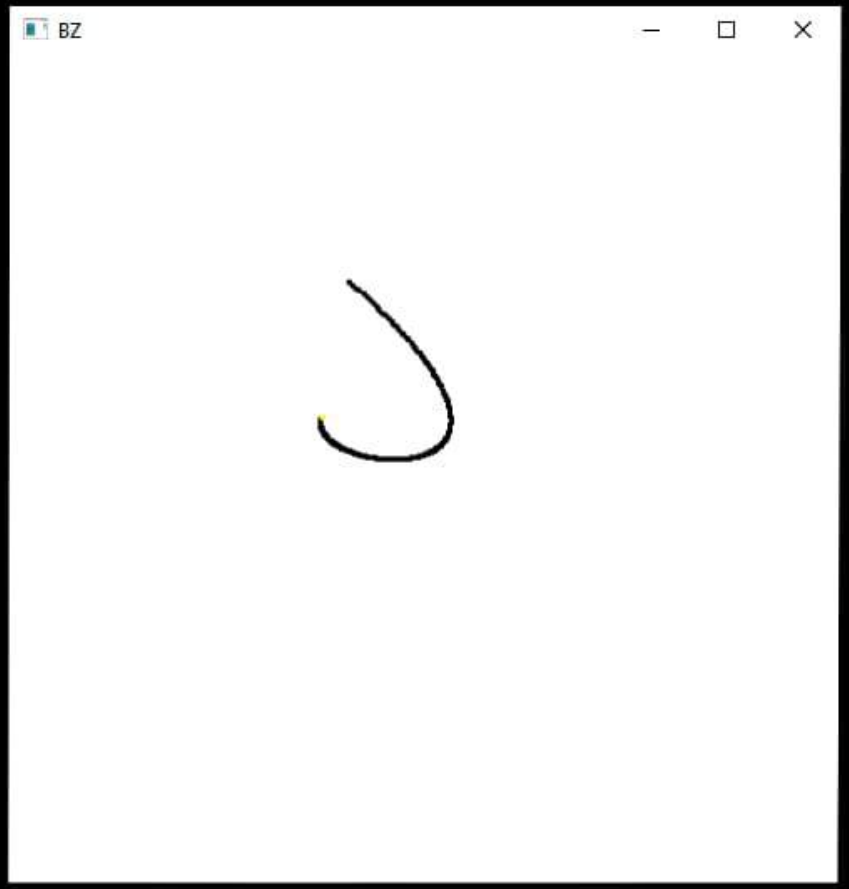
Teacher's Signature _____

```
        1-t,2) * x1[1] + 3 * Pow(t,2) * (1-t) * x1[2] + Pow
            (t,3) * x1[3];
double yt = Pow(1-t,3) * yc[0] + 3 * t * Pow(1-t,2) * yc
    [1] + 3 * Pow(t,2) * (1-t) * yc[2] + Pow(t,3) * yc[3];
glVertex2f(xt, yt);
}
glColor3f(1,1,0);
for(i=0; i<4; i++) {
        glVertex2f(x1[i], yc[i]);
        glEnd();
        glFlush();
}
}
void mymouse(int btn, int state, int x, int y)
{
  if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN &&
                flag<4)
  {
     x1[flag] = x;
     yc[flag] = 500-y;
     cout << "x:" << x << "y:" << 500-y;
     glPointSize(3);
     glColor3f(1,1,0);
     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 ("BZ");
    glutDisplayFunc (display);
    glutMouseFunc (mymouse);
    myInit();
    glutMainLoop();
}
```

C:\Users\Vaibhav\source\repos\labpgm12_BezierCurve\Debug\labpgm12_BezierCurve.exe

```
 X: 203 Y227 X: 272 Y407 X: 227 Y297 X: 227 Y297 X: 227 Y297 X: 227 Y297 X: 227 Y297 X: 227 Y297 X: 382 Y415 X: 297 Y32 X: 249 Y368 X: 249 Y368 X: 249 Y368 X: 208 Y185
X: 358 Y230 X: 202 Y366 X: 188 Y281 X: 179 Y245
```



Type here to search

ENG 14:45 05-12-2020