Name of the Institute ..... RV College of Engineering .....

Laboratory Certificate

This is to certify that ~~smt.~~/Sri ..... Rishab V Arun .....

has satisfactorily completed the course of Experiments in Practical ..... .

..... Computer Graphics .....Prescribed by the

..... VTU .....university

in the Laboratory of this college in the year 20 - 20

Signature of the Teacher in charge of the batch

Date :..... 1/1/21 .....

Head of the Department

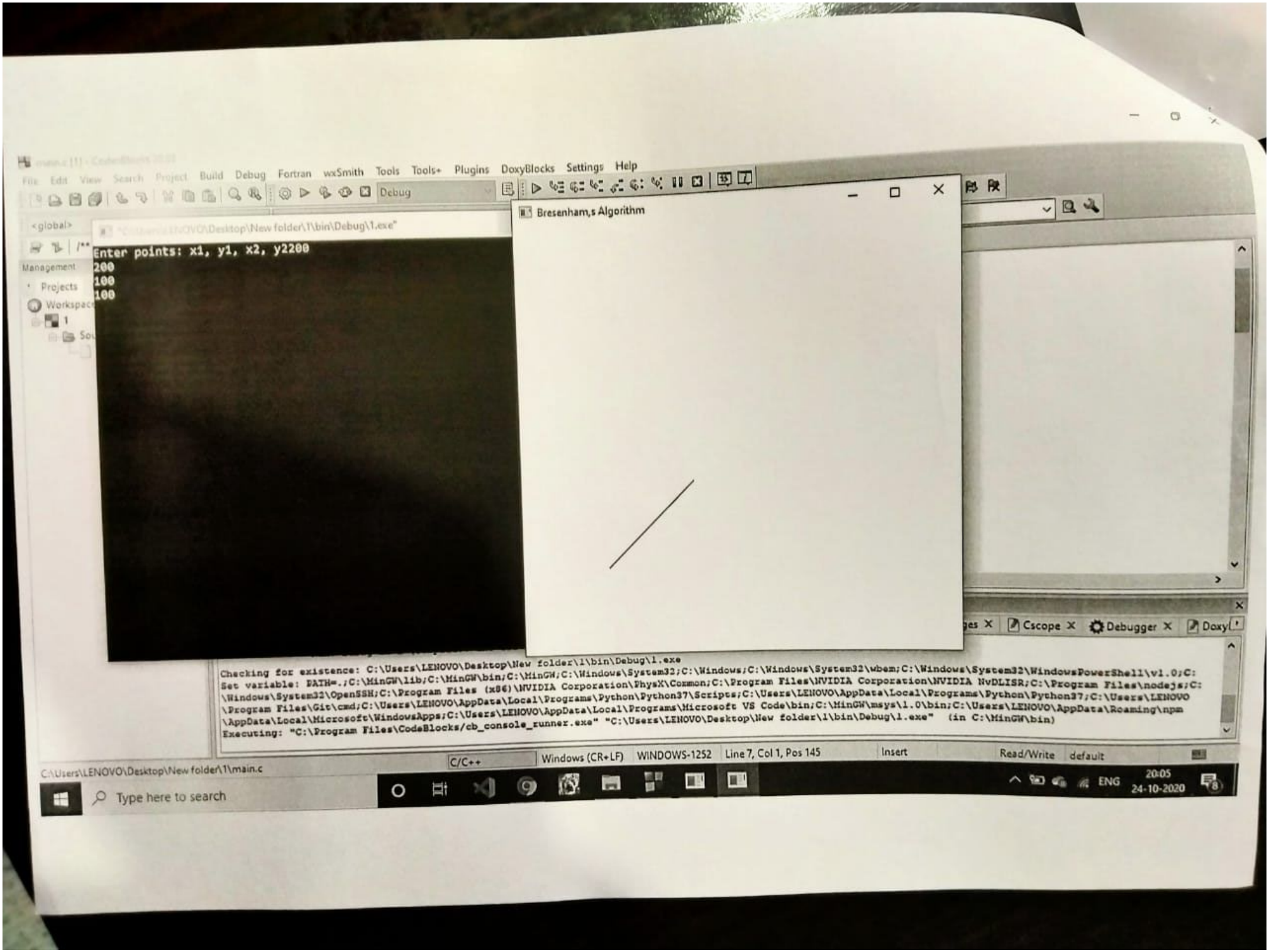
Name of the candidate..... Rishab V Arun .....Reg No..... 1RV17CS122

Examination Centre.....

Date of Practical Examination.....

```
#define BLACK 0
#include <stdio.h>
#include <GL/glut.h>
=
int x1, x2, y1, y2;
void draw_pixel(int x,int y,int value)
{
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
}
void bres(int x1, int x2, int y1, int y2)
{
int dx,dy,i,e;
int incx,incy,inc1,inc2;
int x,y;
dx=x2-x1;
dy=y2-y1;
if(dx<0) dx=-dx;
if(dy<0) dy=-dy;
incx=1;
if(x2<x1)incx=-1;
incy=1;
if(y2<y1) incy=-1;
x=x1;y=y1;
if(dx>dy)
{
draw_pixel(x,y,BLACK);
e=2*dy-dx;
inc1=2*(dy-dx);
inc2=2*dy;
for(i=0;i<dx;i++)
{
if(e>=0)
{
y+=incy;
e+=inc1;
}
else e+=inc2;
x+=incx;
draw_pixel(x,y,BLACK);
}
}
else
{
draw_pixel(x,y,BLACK);
e=2*dx-dy;
inc1=2*(dx-dy);
inc2=2*dx;
for(i=0;i<dy;i++)
{
if(e>=0)
{
x+=incx;
e+=inc1;
}
else e+=inc2;
y+=incy;
draw_pixel(x,y,BLACK);
}
}
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
bres(x1, y1, x2, y2);
glFlush();
}
```

```
void myinit()
{
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,499.0,0.0,499.0);
}
void main(int argc, char** argv)
{

printf("Enter points: x1, y1, x2, y2");
scanf("%d %d %d %d", &x1,&x2,&y1,&y2);

glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Bresenham,s Algorithm");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```


Enter points: x1, y1, x2, y2200
200
100
100
Bresenham,s Algorithm
Checking for existence: C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe" (in C:\MinGW\bin)
C:\Users\LENOVO\Desktop\New folder\1\main.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 7, Col 1, Pos 145
Insert
Read/Write
default
Type here to search
ENG
20:05
24-10-2020

Date .....

Expt. No. 1 Page No. .....

1) Write a program for Bresenham's line drawing method

```
#include <iostream.h>
#include <GL/GLUT.h>
#include <time.h>
using namespace std;
void x1, x2, y1, y2

void draw_pixel (int x, int y)
{  glcolor3f (1,0,0);
   glBegin (GL_POINTS);
   glVertex2i (x,y)
   glEnd();
   glFlush(); }
void Drawline()
{ int dx, dy, i, e, incx, incy, inc2,
  x,y, x2
  dx = x2 - x1;   dy = y2 - y1;
  if (dy < 0) dy = -dy;  if (dx < 0) dx = -dx;
  incx = 1;
  if (x2 < x1) incx = 1;   incy = 1;
  if (y2 < y1) incy = -1;
  x = x1;   y = y1;
  if (dx > dy)  draw_pixel (x, y);
    e = 2*dx - dy;  inc1 = 2*(dx - dy);
    inc2 = 2dy
  for (i=0; i<dx; i++)
    { if (e>0) { y+= incy; e+= inc1; }
```

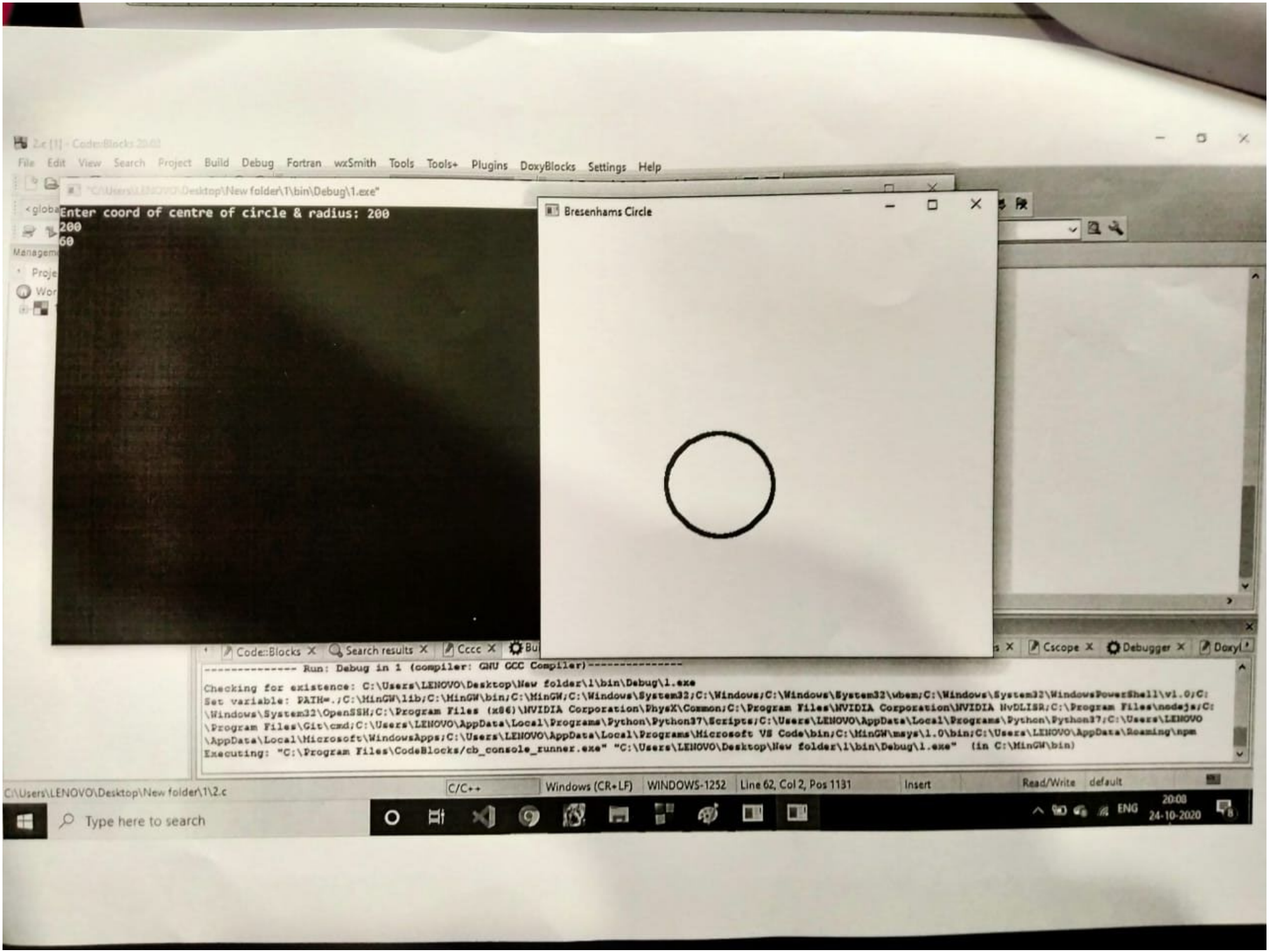
Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
        else    e+=inc2;
                x+=inx;
                draw_pixel(x,y);
            }
        }
    else { drawpixel(x,y); e= 2*dx - dy;
           inc1 = 2(dx-dy); inc2 = 2*dx;
           for (i=0; i<dy; i++){
                if (e>0) { x+=inx; e+=inc1; }
                else e+=inc2;
                y+=iny;
                draw_pixel (x,y); }
               glFlush(); }

void myinit()
  { glClear ( GL_COLOUR_BUFFER_BIT);
    glClearColor(1,1,1,1);
    glOrtho2D( -250, +250 ;250,250); }

void mymouse (int button, int state, int x, int y)
  { switch (button)
      { case GLUT_LEFT_BUTTON:
            if ( state == GLUT_DOWN)
              { if (flag==0) {
                    printf ("Defining x1,y1");
                x1=x-250; y1= 250-y;
                    flag++;
                cout << x1 << " " << y1; }
            else { printf ("Defining x2 y2");
```

Teacher's Signature : _____


File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe"
Enter coord of centre of circle & radius: 200
200
60
Bresenhams Circle
Code::Blocks
Search results
Cccc
Cscope
Debugger
Doxyl
---------- Run: Debug in 1 (compiler: GNU GCC Compiler)----------
Checking for existence: C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe
Set variable: PATH=.;C:\MinGW\lib;C:\MinGW\bin;C:\MinGW;C:\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR;C:\Program Files\nodejs;C:\Program Files\Git\cmd;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37\Scripts;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37;C:\Users\LENOVO\AppData\Local\Microsoft\WindowsApps;C:\Users\LENOVO\AppData\Local\Programs\Microsoft VS Code\bin;C:\MinGW\msys\1.0\bin;C:\Users\LENOVO\AppData\Roaming\npm
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe" (in C:\MinGW\bin)
C:\Users\LENOVO\Desktop\New folder\1\2.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 62, Col 2, Pos 1131
Insert
Read/Write
default
Type here to search
ENG
20:08
24-10-2020

```
#include<stdio.h>
#include<gl/glut.h>
int xc,yc,r;
void drawCircle(int xc,int yc,int x,int y)
{
glBegin(GL_POINTS);
glVertex2i(xc+x,yc+y);
glVertex2i(xc-x,yc+y);
glVertex2i(xc+x,yc-y);
glVertex2i(xc-x,yc-y);
glVertex2i(xc+y,yc+x);
glVertex2i(xc-y,yc+x);
glVertex2i(xc+y,yc-x);
glVertex2i(xc-y,yc-x);
glEnd();
}
void circleBres(int xc,int yc,int r)
{
int x=0,y=r;
int d=3-2*r;
while(x<y)
{
drawCircle(xc,yc,x,y);
x++;
if(d<0)
d=d+4*x+6;
else
{
y--;
d=d+4*(x-y)+10;
}
drawCircle(xc,yc,x,y);
 }
}
void display()
{
int j;
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
circleBres(xc,yc,r);
glFlush();
}
void myinit()
{
glClearColor(1.0,1.0,1.0,0);
glColor3f(0,0,1.0);
glPointSize(5.0);
gluOrtho2D(0.0,500,0.0,500);
}
void main(int argc,char *argv[])
{
int j;
printf("Enter coord of centre of circle & radius: ");
scanf("%d%d%d",&xc,&yc,&r);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,100);
glutCreateWindow("Bresenhams Circle");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

Date .....

Expt. No. 2 Page No. .....

2] Write a program to create Circle & Ellipse.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int xc, yc, r, x, y, xx, yy, xe, ye;
int 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 bres_circle(){
{ glClear(GL_COLOUR_BUFFER_BIT);
  int x=0; y=r; int d=3-2*r;
  while(x<=y) { draw_circle(xc, yc, x, y); x++;
    if(d<0)
      d=d+4*x+6;
    else { y--; d=d+4*(x-y)+10; }
    draw_circle(xc, yc, x, y); glFlush();
  }
int p1x, p2x, p1y, p2y, points_done=0;
void draw_ellipse (int xcc, int ycc, int x, int y)
{ glBegin(GL_POINTS);
  glVertex2i(x+xc, y+yc); glVertex2i(-x+xc, y+yc);
  glVertex2i(x+xc, -y+yc); glVertex2i(-x+xc, -y+yc);
  glEnd();
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void mid-ellipse()
{ glClear (GL_COLOR_BUFFER_BIT)
  float dx, dy, d1, d2, x, y, x=0, y=ry;
  d1 = (ry*ry) - (rx*rx*rx) + (0.25*rx
                                *rx);
  dx = 2*ry*ry*x;
  dy = 2*rx*ry*y;
  while (dx < dy)
  { draw_ellipse (xc, yc, 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 (xc, yc, 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*ry);
      d2 = d2 + dx - dy + (rx*rx); }
    glFlush();
  }
}
void main (int argc, char* argv[])
{ glutInit (&argc, argv);
```

Teacher's Signature : _____

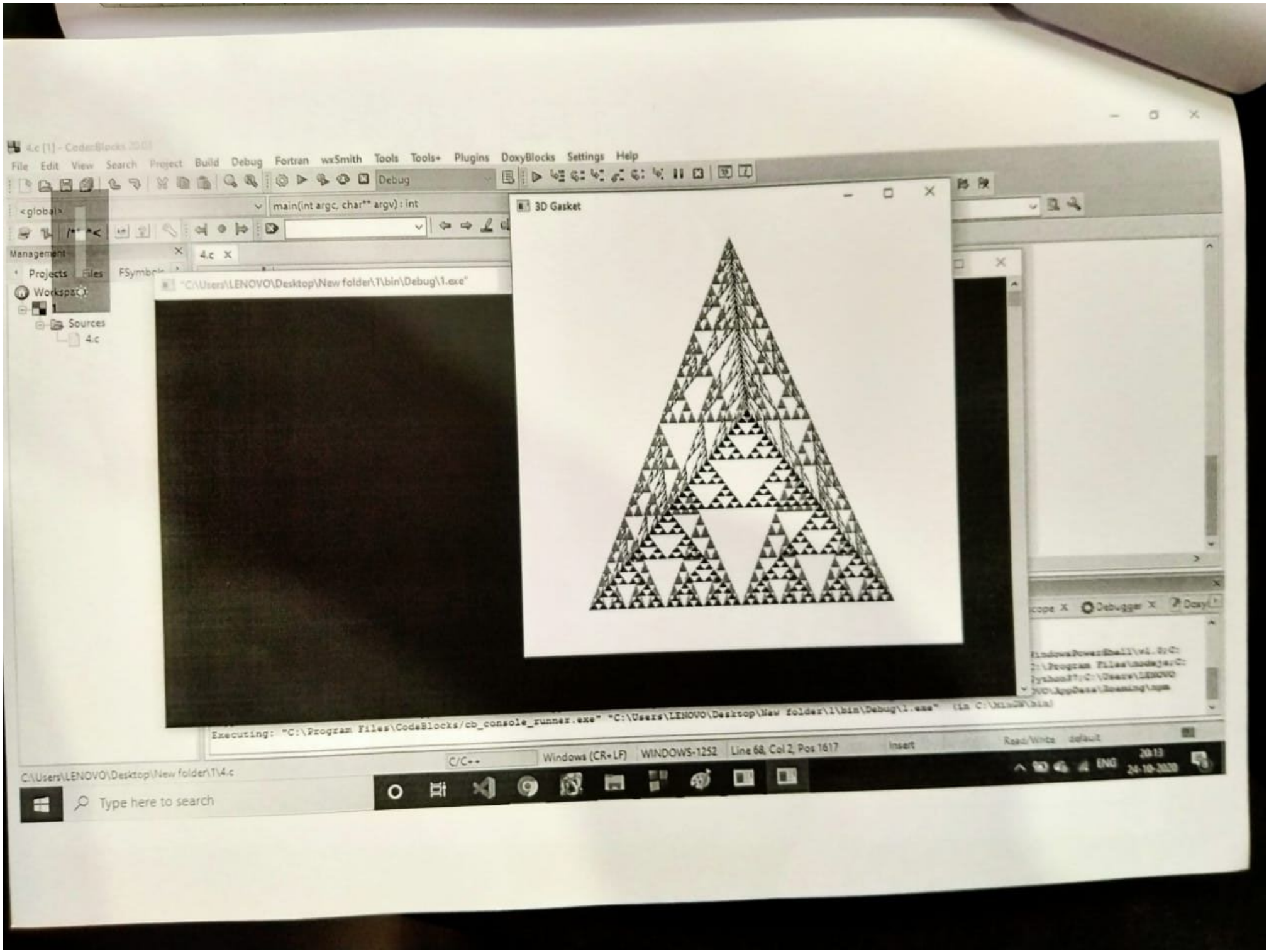
Date .....

Expt. No. .....

Page No. .....

```
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (0,0);
printf ("enter 1 for circle 2 for ellipse");
int ch; scanf ("%d", &ch);
switch (ch) {
 case 1: printf ("Enter circle parameters");
         scanf ("%d %d %d", &xc, &yc, &r)
         glutCreateWindow ("Circle");
         glutDisplayFunc (draw_circle); break;
 case 2: printf ("Enter ellipse parameters");
         scanf ("%d %d %d %d", &xc, &yc
                &rx, &ry);
         glutCreateWindow ("Ellipse");
         glutDisplayFunc (midEllipse); break;
 }
 myInit();
 glutMainLoop();
}
```

Teacher's Signature : _____


3D Gasket
"C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe"
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\New folder\1\bin\Debug\1.exe"
Windows (CR+LF)
WINDOWS-1252
Line 68, Col 2, Pos 1617
Insert
Read/Write
default
C/C++
Type here to search

```
#include <GL/glut.h>
/* initial triangle */
typedef GLfloat point[3];
point v[]={{30.0, 50.0, 100.0}, {0.0, 450.0, -150.0},
 {-350.0, -400.0, -150.0}, {350., -400., -150.0}};
int n; /* number of recursive steps */
void triangle( point a, point b, point c)
/* display one triangle */
{
 glBegin(GL_TRIANGLES);
 glVertex3fv(a);
 glVertex3fv(b);
 glVertex3fv(c);
glEnd();
}
void divide_triangle(point a, point b, point c, int m)
{
/* triangle subdivision using vertex numbers */
 point v0, v1, v2;
 int j;
 if(m>0)
 {
 for(j=0; j<3; j++) v0[j]=(a[j]+b[j])/2;
 for(j=0; j<3; j++) v1[j]=(a[j]+c[j])/2;
 for(j=0; j<3; j++) v2[j]=(b[j]+c[j])/2;
 divide_triangle(a, v0, v1, m-1);
 divide_triangle(c, v1, v2, m-1);
 divide_triangle(b, v2, v0, m-1);
 }
 else(triangle(a,b,c));
/* draw triangle at end of recursion */
}
void tetra(int m)
{
glColor3f(1.0,0.0,0.0);
divide_triangle(v[0],v[1],v[2],m);
glColor3f(0.0,1.0,0.0);
divide_triangle(v[3],v[2],v[1],m);
glColor3f(0.0,0.0,1.0);
divide_triangle(v[0],v[3],v[1],m);
glColor3f(0.0,0.0,0.0);
divide_triangle(v[0],v[2],v[3],m);
}
void display()
{
glClearColor (1.0, 1.0, 1.0,1.0);
 glClear(GL_COLOR_BUFFER_BIT);
 tetra(n);
 glFlush();
}
void myinit()
{
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 glOrtho(-499.0, 499.0, -499.0, 499.0,-499.0,499.0);
 glMatrixMode(GL_MODELVIEW);
 }
int main(int argc, char **argv)
{
 n=5;
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitWindowSize(500, 500);
 glutCreateWindow("3D Gasket");
 glutDisplayFunc(display);
myinit();
 glutMainLoop();
 }
```

Date .....

Expt. No. 3 Page No. .....

3) Write a program that recursively subdivides a tetrahedron to form 3D sierpinski gasket

```
#include <GL/glut.h>
#include <iostream.h>
using namespace std; int n;
float tetro[4][3] = {{0,200,-400},{0,0,-250},{200,-350,-300},{-200,-300,-200}};
void draw_triangle (float p1[], float p2[], float p3[])
{ glBegin (GL_TRIANGLES)
  glVertex3f (p1[0], p1[1], p1[2]);
  glVertex3f (p2[0], p2[1], p2[2]);
  glVertex3f (p3[0], p3[1], p3[2]);
  glEnd ();
}
void divide_triangle (float a[], float b[], float c[], int m)
{ float v1[3], v2[3], v3[3], int j;
  if (m>0)
  { for (j=0; j<3; j++) v1[j] = ((a[j]+b[j])/2);
    for (j=0; j<3; j++) v2[j] = ((a[j]+c[j])/2);
    for (j=0; j<3; j++) v3[j] = ((b[j]+c[j])/2);
    divide_triangle (a, v1, v2, m-1);
    divide_triangle (b, v3, v1, m-1);
    divide_triangle (c, v2, v3, m-1);
  }
  else draw_triangle (a,b,c);
}
void tetrahedron () { glClear (GL_COLOR_BUFFER_BIT);
```

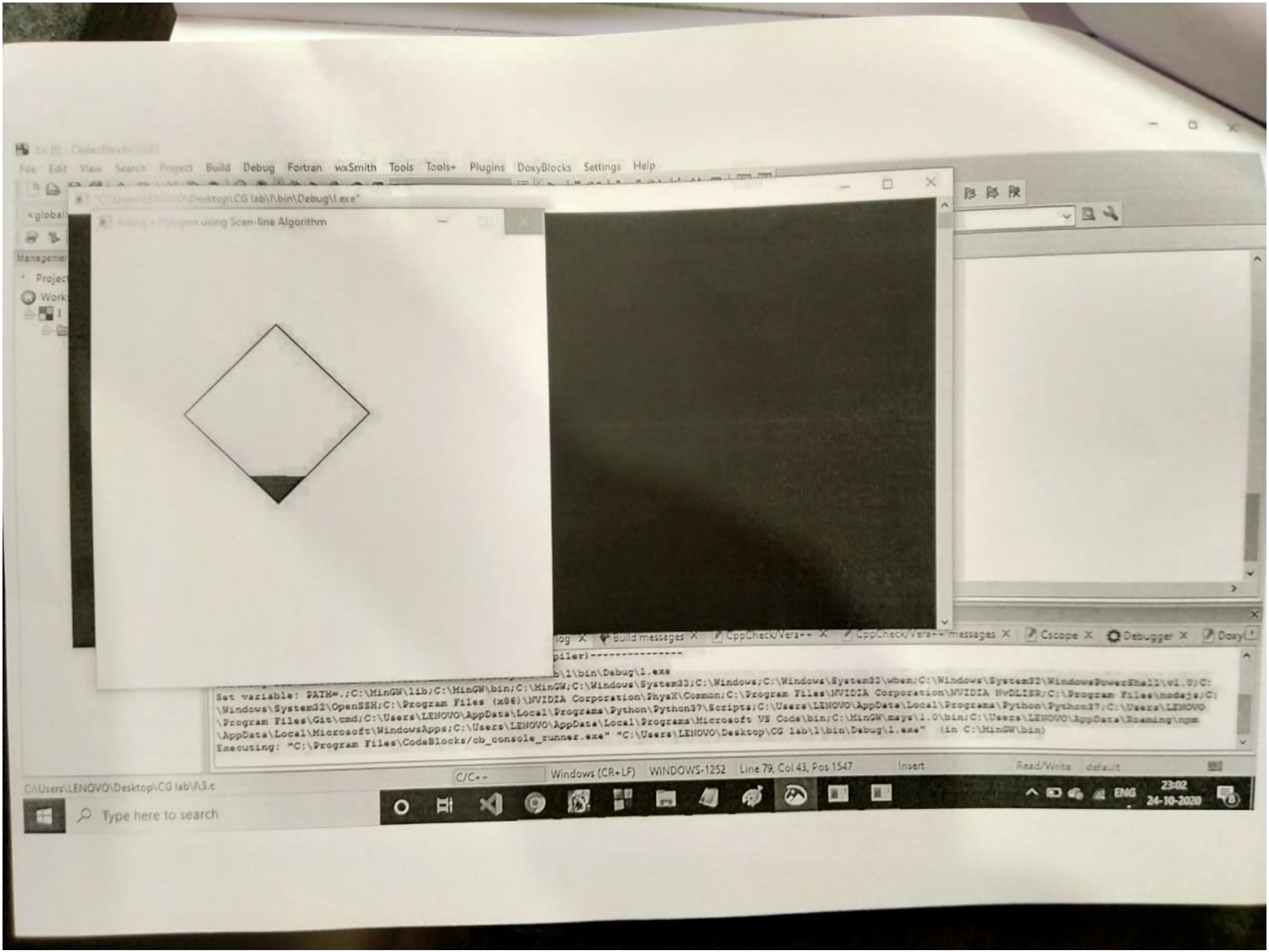
Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
divide_triangle (tetra[0], tetra[1], tetra[2], n);
divide_triangle (tetra[3], tetra[2], tetra[1], n);
divide_triangle (tetra[0], tetra[3], tetra[1], n);
divide_triangle (tetra[0], tetra[2], tetra[3], n);
glEnd() }
void minit() { glClearColor(1,1,1,0); glColor3f(1,0,0);
glPointSize(5.0); glOrtho(-500,500,-500,500,-1,1);
void main(int argc, char* argv[])
{ cout << "Enter n";
  cin >> n;
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE);
  glutInitWindowSize(300,300);
  glutInitWindowPosition(100,100);
  glutCreateWindow("Display");
  glutDisplayFunc(Tetrahedron);
  glEnable(GL_DEPTH_TEST);
  minit();
  glutMainLoop();
}
```

Teacher's Signature : _____


Filling a Polygon using Scan-line Algorithm
Build messages
CppCheck/Vera++
Cscope
Debugger
Set variable: PATH=.;C:\MinGW\lib;C:\MinGW\bin;C:\MinGW;C:\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:
\Windows\System32\OpenSSH;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR;C:\Program Files\nodejs;C:
\Program Files\Git\cmd;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37\Scripts;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37;C:\Users\LENOVO
\AppData\Local\Microsoft\WindowsApps;C:\Users\LENOVO\AppData\Local\Programs\Microsoft VS Code\bin;C:\MinGW\msys\1.0\bin;C:\Users\LENOVO\AppData\Roaming\npm
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\CG lab\1\bin\Debug\1.exe" (in C:\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 79, Col 43, Pos 1547
Insert
Read/Write
default
Type here to search
ENG
23:02
24-10-2020

```
#include <GL/glut.h>
#include<windows.h>
float x1,x2,x3,x4,y1,y2,y3,y4;
void edgedetect(float x1,float y1,float x2,float y2,int *le,int *re)
{
float mx,x,temp;
int i;
if((y2-y1)<0)
{
temp=y1;y1=y2;y2=temp;
temp=x1;x1=x2;x2=temp;
}
if((y2-y1)!=0)
mx=(x2-x1)/(y2-y1);
else mx=x2-x1;
x=x1;
for(i=y1;i<=y2;i++)
{
if(x<(float)le[i])
le[i]=(int)x;
if(x>(float)re[i])
re[i]=(int)x;
x+=mx;
}
}
void draw_pixel(int x,int y)
{
glColor3f(1.0,0.0,0.0);
Sleep(10); // To set the delay time
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
glFlush();
}
void scanfill(float x1,float y1,float x2,float y2,float x3,float y3,float
x4,float y4)
{
int le[500],re[500],i,y;
for(i=0;i<500;i++)
{
le[i]=500;
re[i]=0;
}
edgedetect(x1,y1,x2,y2,le,re);
edgedetect(x2,y2,x3,y3,le,re);
edgedetect(x3,y3,x4,y4,le,re);
edgedetect(x4,y4,x1,y1,le,re);
for(y=0;y<500;y++)
{
if(le[y]<=re[y])
for(i=(int)le[y];i<(int)re[y];i++)
draw_pixel(i,y);
}
}
void display()
{
x1=200.0;y1=200.0;x2=100.0;y2=300.0;x3=200.0;y3=400.0;x4=300.0;y4=300.0;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
 glVertex2f(x1,y1);
 glVertex2f(x2,y2);
 glVertex2f(x3,y3);
 glVertex2f(x4,y4);
glEnd();
scanfill(x1,y1,x2,y2,x3,y3,x4,y4);
glFlush();
}
```

```
void myinit()
{
glclearcolor(1.0,1.0,1.0,1.0);
glcolor3f(1.0,0.0,0.0);
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);
glutCreateWindow("Filling a Polygon using Scan-line Algorithm");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

Date .....

Expt. No. 4 Page No. .....

4) Write a program to fill a polygon using scan filling algorithm

```
#include <GL/glut.h>
#include <iostream>
#include <algorithm>
#include <windows>
using namespace std;
float x[100], y[100]; int n, m; int wx=500, wy=500;
static float int x[10] = {};
void drawline (float x1, float y1, float x2, float y2)
{ sleep (10);
  glColor3f (1,0,0); glBegin (GL_LINES); glVertex2f
  (x1, y1); glVertex2f (x2, y2); glEnd();
}
void Edgedetect (float x1, float y1, float x2, float y2)
{ float temp;
  int temp; if (y2<y1) {temp = x1; x2=y2; x2=temp;}
  if (scanline >y1 && scanline <y2)
    int [m++] = x + (scanline -y1) * (x2-y1)/(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], &1); }
  sort (int x, (n+x+m));
  if (m>=2) for(int i=0 ; i<m ; i=i+2)
    {drawline (int x[i]), &s1, int x[i+1], &s1); }}
void display_filled_polygon(){
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
glClear (GL_COLOR_BUFFER_BIT); glLineWidth (2);
glBegin (GL_LINE_LOOP); for (int i=0; i<n; i++)
  glVertex2f(x[i], y[i]); glEnd(); scanf (x,y); }

void myinit() { glClearColor (1,1,1); glColor3f(0,0,1);
glPointSize (5); gluOrtho2D (0, wx, 0, wy);
}

void main ( int argc, char** argv)
{ glutInit (&argc, argv);
  printf ("Enter Nodes"); scanf ("%d", &n);
  for (i=0; i<n; i++)
    { printf (" x, y");
      scanf ("%f %f", &x[i], &y[i]); }
  glutInitDisplayMode( GLUT_SINGLE);
  glutInitWindowSize ( 500, 500);
  glutCreateWindow ("Scanline");
  glutDisplayFunc( display_filled_polygon);
  myinit(); glutMainLoop();
}
```

Teacher's Signature : _____

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


//RIGHT CLICK TO SHOW REFLECTED HOUSE


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();
        //NORMAL HOUSE
        glColor3f(1, 0, 0);
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 11; i++)
                glVertex2fv(house[i]);
        glEnd();
        glFlush();
        //ROTATED HOUSE
        glPushMatrix();
        glTranslatef(100, 100, 0);
        glRotatef(angle, 0, 0, 1);
        glTranslatef(-100, -100, 0);
        glColor3f(1, 1, 0);
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 11; i++)
                glVertex2fv(house[i]);
        glEnd();
        glPopMatrix();
        glFlush();
}
void display2()
{
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-450, 450, -450, 450);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        //normal house
        glColor3f(1, 0, 0);
        glBegin(GL_LINE_LOOP);
        for (int i = 0; i < 11; i++)
                glVertex2fv(house[i]);
        glEnd();
        glFlush();
        // line
        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();
```

Date .....

Expt. No. ..... Page No. .....

```
            x2, y2, ix, iy, Kx, Ky); new_poly_size++;
        } else {
    } poly_size = new_poly_size;
        for (int i=0; i< poly_size; i++)
        {
            poly_points[i][0] = new_points[i][0];
            poly_points[i][i] = new_points[i][1];
        }
}
void init() { glClearColor(0,0,0,0);
    glMatrixMode(GL_PROJECTION); glLoadIdentity();
        gluOrtho2D(0,500,0,500,0,500);
        glClear(GL_COLOR_BUFFER_BIT);
}
void display() { int i; glColor3f(1,0,0);
    drawPoly(clipper_points, clipper_size);
    glColor3f(0,1,0);
    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[k][0], clipper_points[k][1]);
}
    glColor3f(0,0,1);
    drawPoly(poly_points, poly_size);
    glFlush();
}
int main(int ac, char** av[])
```

Teacher's Signature : _____

```
        for (int i = 0; i < clipper_size; i++)
        {
                int k = (i + 1) % clipper_size;

                // We pass the current array of vertices, it's size
                // and the end points of the selected clipper line
                clip(poly_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();
}

//Driver code
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("%d%d", &poly_points[i][0], &poly_points[i][1]);
                org_poly_points[i][0] = poly_points[i][0];
                org_poly_points[i][1] = poly_points[i][1];
        }

        printf("Enter no. of vertices of clipping window:");
        scanf("%d", &clipper_size);
        for (int i = 0; i < clipper_size; i++)
        {
                printf("Clip Vertex:\n");
                scanf("%d%d", &clipper_points[i][0], &clipper_points[i][1]);
        }

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(400, 400);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("Polygon Clipping!");
        glutDisplayFunc(display);
        glutMainLoop();
        return 0;
}
```

```
        glFlush();
}
void myinit()
{
        glClearColor(1, 1, 1, 1);
        glColor3f(1, 0, 0);
        glPointSize(1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("clip");
        myinit();
        glutDisplayFunc(display);
        glutMainLoop();
}
```

Date .....

Page No. .....

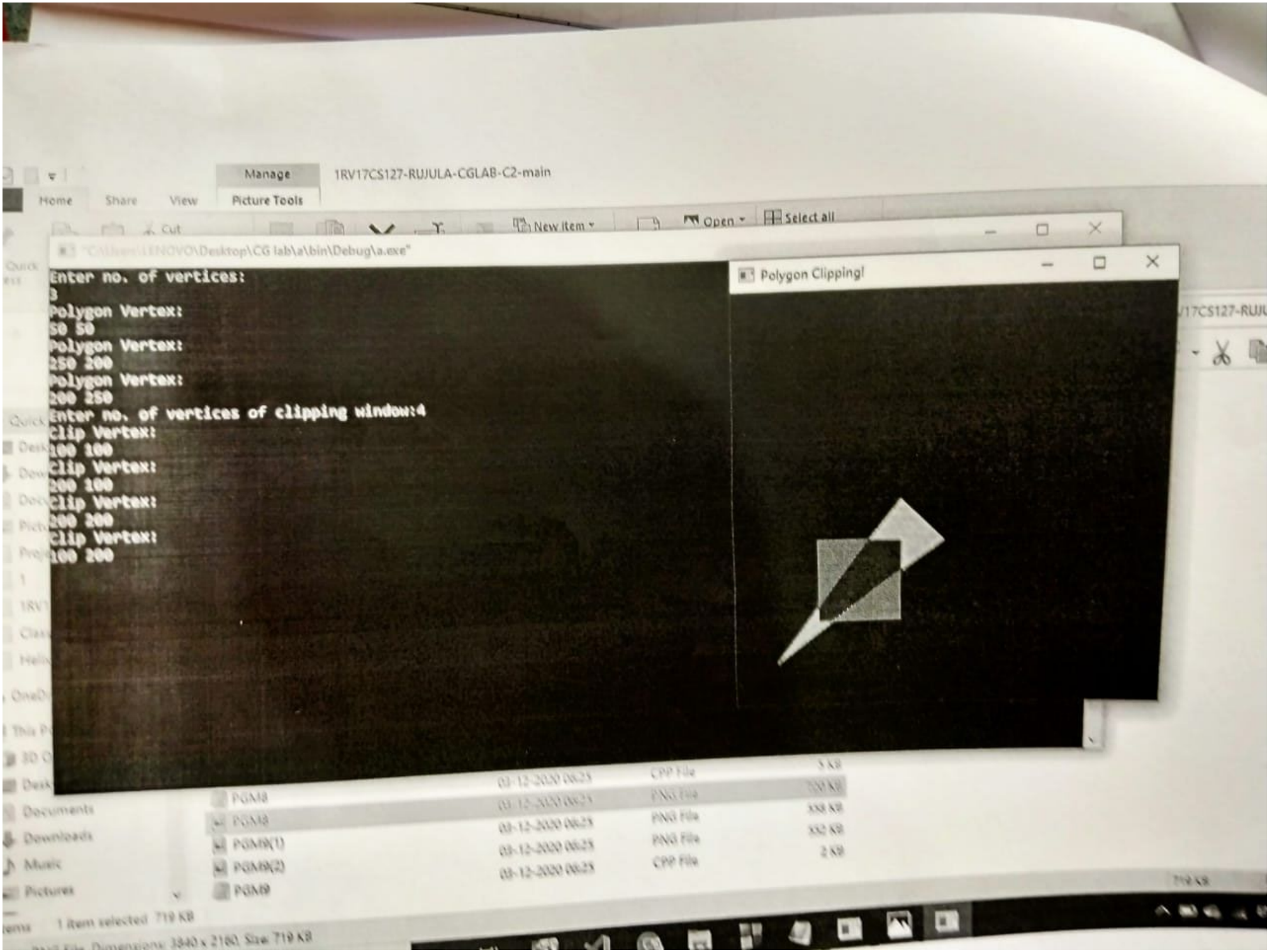
Expt. No. .....

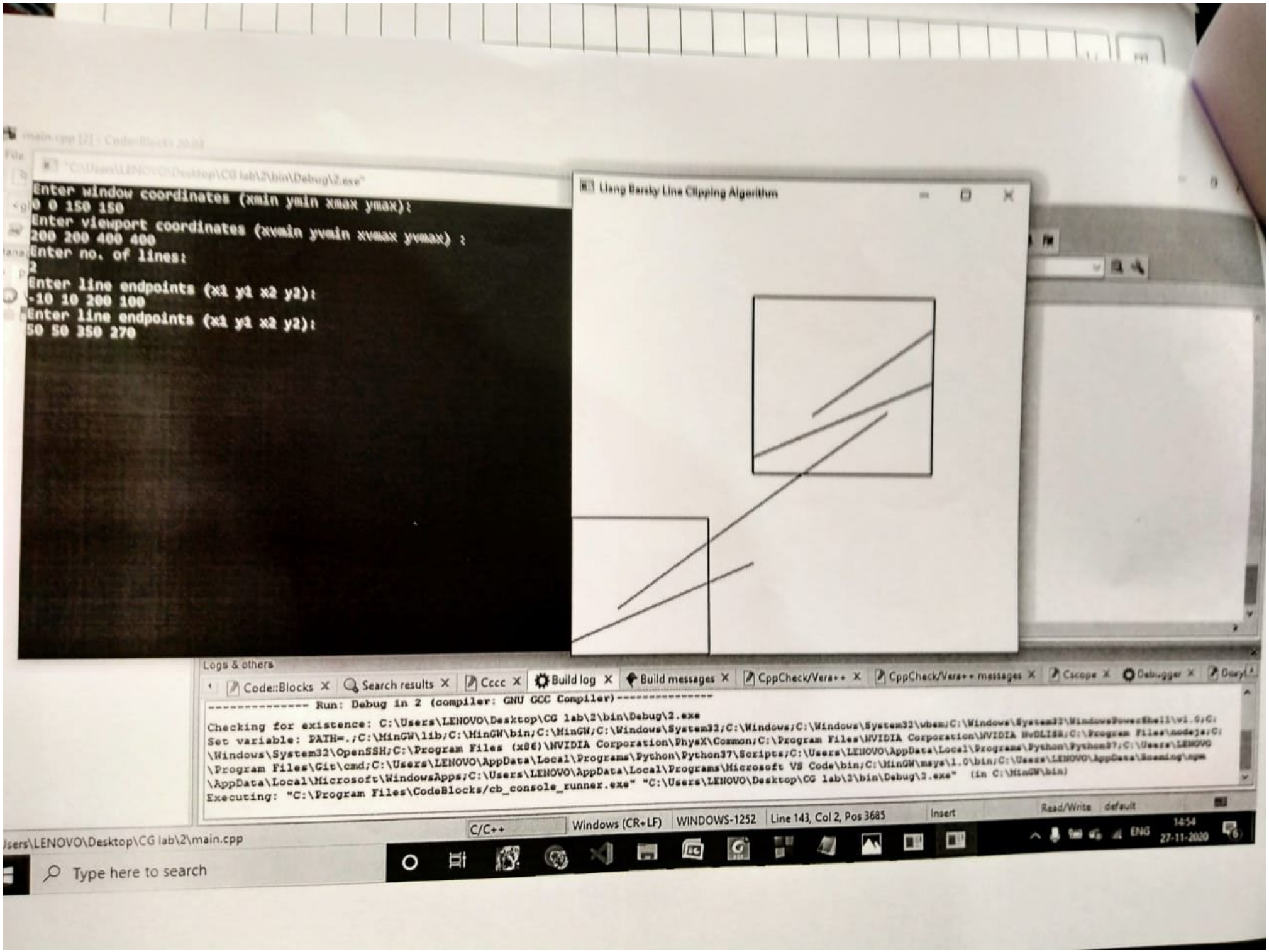
```
void display() { glClear(GL_COLOR_BUFFER_BIT)
  { glColor3f(0,0,1); glBegin(GL_LINE_LOOP);
    glVertex2f(xmin, ymin);
    glVertex2f(xmax, ymin);
    glVertex2f(xmax, ymax);
    glVertex2f(xmin, ymax); glEnd();
  for(int i=0; i<n; i++) { glBegin(GL_LINES);
    glVertex2d(ls[i].x1, ls[i].y1);
    glVertex2d(ls[i].x2, ls[i].y2); glEnd(); glFlush();
}

void myinit() { glClearColor(1,1,1,1); glPointSize(1.0);
  glMatrixMode(GL_PROJECTION); glLoadIdentity();
  glOrtho2D(0,500,0,500); }

void main(int ac, char **av) {
  glutInit(&ac, &av);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(500,500); glutInitWindowPos(100,100);
  glutCreateWindow("Clip");
  printf("Enter co-ordinates");
  scanf("%f %f %f %f", &xmin, &ymin, &xmax, &ymax);
  scanf("%f %f %f %f", &xvmin, &yvmin, &xvmax, &yvmax);
  printf("Enter lines");
  scanf("%d", &n);
  for(int i=0; i<n; i++) scanf("%f %f %f %f",
&ls[i].x1, &ls[i].x2, &ls[i].y1, &ls[i].y2);
  glutDisplayFunc(display);
  glutMainLoop(); }
```

Teacher's Signature : _____

```
#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);
        glEndList();
}
void mykeyboard(unsigned char key, int x, int y) {
        switch (key) {
        case 't': glutPostRedisplay();
                break;
        case 'q': exit(0);
        default: break;

        }
}

void myInit() {
        glClearColor(0, 0, 0, 0);
        glOrtho(0, 600, 0, 600, 0, 600);

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

}

void moveCar(float s) {
        glTranslatef(s, 0.0, 0.0);
        glCallList(CAR);
        glPushMatrix();
        glTranslatef(25, 25, 0.0);      //move to first wheel position
        //draw_wheel();
        glCallList(WHEEL);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(75, 25, 0.0);     //move to 2nd wheel position
        ////draw_wheel();
        glCallList(WHEEL);
        glPopMatrix();
        glFlush();
}
void myDisp() {
        glClear(GL_COLOR_BUFFER_BIT);
        carlist();
        moveCar(s);
        wheellist();
```

```
// C++ program for implementing Sutherland–Hodgman
// algorithm for polygon clipping
#include<iostream>
#include<GLUT/glut.h>
using namespace std;
int poly_size, poly_points[20][2], org_poly_size, org_poly_points[20][2],
clipper_size, clipper_points[20][2];
const int MAX_POINTS = 20;


// Returns x-value of point of intersection of two
// lines

void drawPoly(int p[][2], int n) {
        glBegin(GL_POLYGON);
        for (int i = 0; i < n; i++)
                glVertex2f(p[i][0], p[i][1]);
        glEnd();
}

int x_intersect(int x1, int y1, int x2, int y2,
        int x3, int y3, int x4, int y4)
{
        int num = (x1 * y2 - y1 * x2) * (x3 - x4) -
                (x1 - x2) * (x3 * y4 - y3 * x4);
        int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
        return num / den;
}

// Returns y-value of point of intersectipn of
// two lines
int y_intersect(int x1, int y1, int x2, int y2,
        int x3, int y3, int x4, int y4)
{
        int num = (x1 * y2 - y1 * x2) * (y3 - y4) -
                (y1 - y2) * (x3 * y4 - y3 * x4);
        int den = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4);
        return num / den;
}

// This functions clips all the edges w.r.t one clip
// edge of clipping area
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;

        // (ix,iy),(kx,ky) are the co-ordinate values of
        // the points
        for (int i = 0; i < poly_size; i++)
        {
                // i and k form a line in polygon
                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];

                // Calculating position of first point
                // w.r.t. clipper line
                int i_pos = (x2 - x1) * (iy - y1) - (y2 - y1) * (ix - x1);

                // Calculating position of second point
                // w.r.t. clipper line
                int k_pos = (x2 - x1) * (ky - y1) - (y2 - y1) * (kx - x1);

                // Case 1 : When both points are inside
                if (i_pos >= 0 && k_pos >= 0)
                {
                        //Only second point is added
```


Manage
1RV17CS127-RUJULA-CGLAB-C2-main
Home
Share
View
Picture Tools
New item
Open
Select all
"C:\Users\LENOVO\Desktop\CG lab\a\bin\Debug\a.exe"
Enter no. of vertices:
3
Polygon Vertex:
50 50
Polygon Vertex:
250 200
Polygon Vertex:
200 250
Enter no. of vertices of clipping window:4
Clip Vertex:
100 100
Clip Vertex:
200 100
Clip Vertex:
200 200
Clip Vertex:
100 200
Polygon Clipping!
Documents
Downloads
Music
Pictures
PGM8
PGM8
PGM8(1)
PGM8(2)
PGM9
1 item selected 719 KB


Enter window coordinates (xmin ymin xmax ymax):
0 0 150 150
Enter viewport coordinates (xvmin yvmin xvmax yvmax) :
200 200 400 400
Enter no. of lines:
2
Enter line endpoints (x1 y1 x2 y2):
-10 10 200 100
Enter line endpoints (x1 y1 x2 y2):
50 50 350 270
Liang Barsky Line Clipping Algorithm
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
---------- Run: Debug in 2 (compiler: GNU GCC Compiler)----------
Checking for existence: C:\Users\LENOVO\Desktop\CG lab\2\bin\Debug\2.exe
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\CG lab\2\bin\Debug\2.exe"
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 143, Col 2, Pos 3685
Insert
Read/Write
default
Type here to search
ENG
14:54
27-11-2020

```
#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;

outcode computeoutcode(double x, double y)
{
        outcode code = 0;
        if (y > ymax)
                code |= TOP;
        else if (y < ymin)
                code |= BOTTOM;
        if (x > xmax)
                code |= RIGHT;
        else if (x < xmin)
                code |= LEFT;

        return code;
}

void cohensuther(double x0, double y0, double x1, double y1)
{
        outcode 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
```

Date .....

Expt. No. ..... Page No. .....

```
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) { x0 = x0+u1*dx;
                      y0 = y0+u1*dy; }
          double sx = (xvmax-xvmin)/(xmax-xmin);
          double sy = (yvmax-yvmin)/(ymax-ymin);
          glColor3f(0,0,1);
          glBegin(GL_LINES);
          glVertex2d(vx0, vx0);
          glVertex2d(vx1, vy1);
          glEnd();
        }
}
void display(){ glClear(GL_COLOR_BUFFER_BIT);
  for(i=0; i<n; i++){
    { glBegin(GL_LINES);
      glVertex2d(ls[i].x1, ls[i].y1);
      glVertex2d(ls[i].x2, ls[i].y2);
    } glEnd(); }
  glColor3f(0,1,0);
  glBegin(GL_LINE_LOOP) glVertex2f(xmin, ymin);
  glVertex2f(xmax, ymin); glVertex2f(xmax, xmax);
  glVertex2f(xmin, xmax); glEnd();
  for(int i=0; i<n; i++)
```

Teacher's Signature : _____

```
                new_points[new_poly_size][0] = kx;
                new_points[new_poly_size][1] = ky;
                new_poly_size++;
        }

        // Case 2: When only first point is outside
        else if (i_pos < 0 && k_pos >= 0)
        {
                // Point of intersection with edge
                // and the second point is added
                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++;
        }

        // Case 3: When only second point is outside
        else if (i_pos >= 0 && k_pos < 0)
        {
                //Only point of intersection with edge is added
                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++;
        }

        // Case 4: When both points are outside
        else
        {
                //No points are added
        }
    }

    // Copying new points into original array
    // and changing the no. of vertices
    poly_size = new_poly_size;
    for (int i = 0; i < poly_size; i++)
    {
            poly_points[i][0] = new_points[i][0];
            poly_points[i][1] = new_points[i][1];
    }
}


void init() {
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 500.0, 0.0, 500.0, 0.0, 500.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

// Implements Sutherland♦Hodgman algorithm
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);
    //i and k are two consecutive indexes
```

```
        //Reflected
        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++)
                glVertex2fv(house[i]);
        glEnd();
        glPopMatrix();
        glFlush();
}
void myInit() {
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glColor3f(1.0, 0.0, 0.0);
        glLineWidth(2.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-450, 450, -450, 450);
}
void mouse(int btn, int state, int x, int y) {
        if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
                display();
        }
        else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {
                display2();
        }
}
void main(int argc, char** argv)
{
        printf("Enter the rotation angle\n");
        scanf_s("%d", &angle);
        printf_s("Enter c and 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();
}
```

Date .....

Expt. No. 6 Page No. .....

6) Write a program for Cohen - Sutherland algorithm

```
#include <stdio.h>  #include <stdlib.h>  #include <graphics.h>
#define OUTCODE int  #define true 1  #define FALSE 0
double xmin, ymin, xmax, ymax;
double xvmin, yvmin, xvmax, yvmax;
const int RIGHT = 4, TOP = 1, LEFT = 8, BOTTOM = 2;
int n; struct line_segment { int x1, y1, x2, y2; };
struct line_segment ls[10];
Outcode compute_code ( double x, double y)
{   outcode code = 0;
    if (y > ymax) code |= TOP;
    else if ( y < ymin ) code |= BOTTOM;
    if ( x > xmax) code |= RIGHT;
     else if ( x < xmin) code |= LEFT;
    return code;
}
void cohen_sutherland ( double x0, double y0, double x1,
double y1) { outcode oc0, oc1, oc;
   bool accept = false, done = false;
oc0 = computecode (x0, y0);
oc1 = compute code ( x1, y1);
do { if ( !(oc0 | oc1) ) { accept = true; done = true; }
else if ( oc0 & oc1) done = true;
else { double x, y;
        oc = oc0 ? oc0 : oc1;
if (outcode & TOP) { x = x0 + (x1 - x0) * (ymax
                        - y0) / (y1 - y0); y = ymax;
                    }
```

Teacher's Signature : _____

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

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

int n;

struct line_segment {
        int x1;
        int y1;
        int x2;
        int y2;
};
struct line_segment ls[10];
int cliptest(double p, double q, double* u1, double* u2)
{
        double r;
        if (p) r = q / p;  // to check whether p
        if (p < 0.0)    // potentially entry point, update te
        {
                if (r > * u1)*u1 = r;
                if (r > * u2) return(false); // line portion is outside
        }
        else
                if (p > 0.0)    //  Potentially leaving point, update tl
                {
                        if (r < *u2)*u2 = r;
                        if (r < *u1) return(false); // line portion is outside
                }
                else
                        if (p == 0.0)
                        {
                                if (q < 0.0) return(false); // line parallel to
edge but outside
                        }
        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;
        //draw a red colored viewport
        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))  // inside test wrt left edge
                if (cliptest(dx, xmax - x0, &u1, &u2)) // inside test wrt right
edge
                        if (cliptest(-dy, y0 - ymin, &u1, &u2)) // inside test
wrt bottom edge
                                if (cliptest(dy, ymax - y0, &u1, &u2)) // inside
test wrt top edge
                                {
                                        if (u2 < 1.0)
                                        {
                                                x1 = x0 + u2 * dx;
                                                y1 = y0 + u2 * dy;
                                        }
                                        if (u1 > 0.0)
                                        {
                                                x0 = x0 + u1 * dx;
```

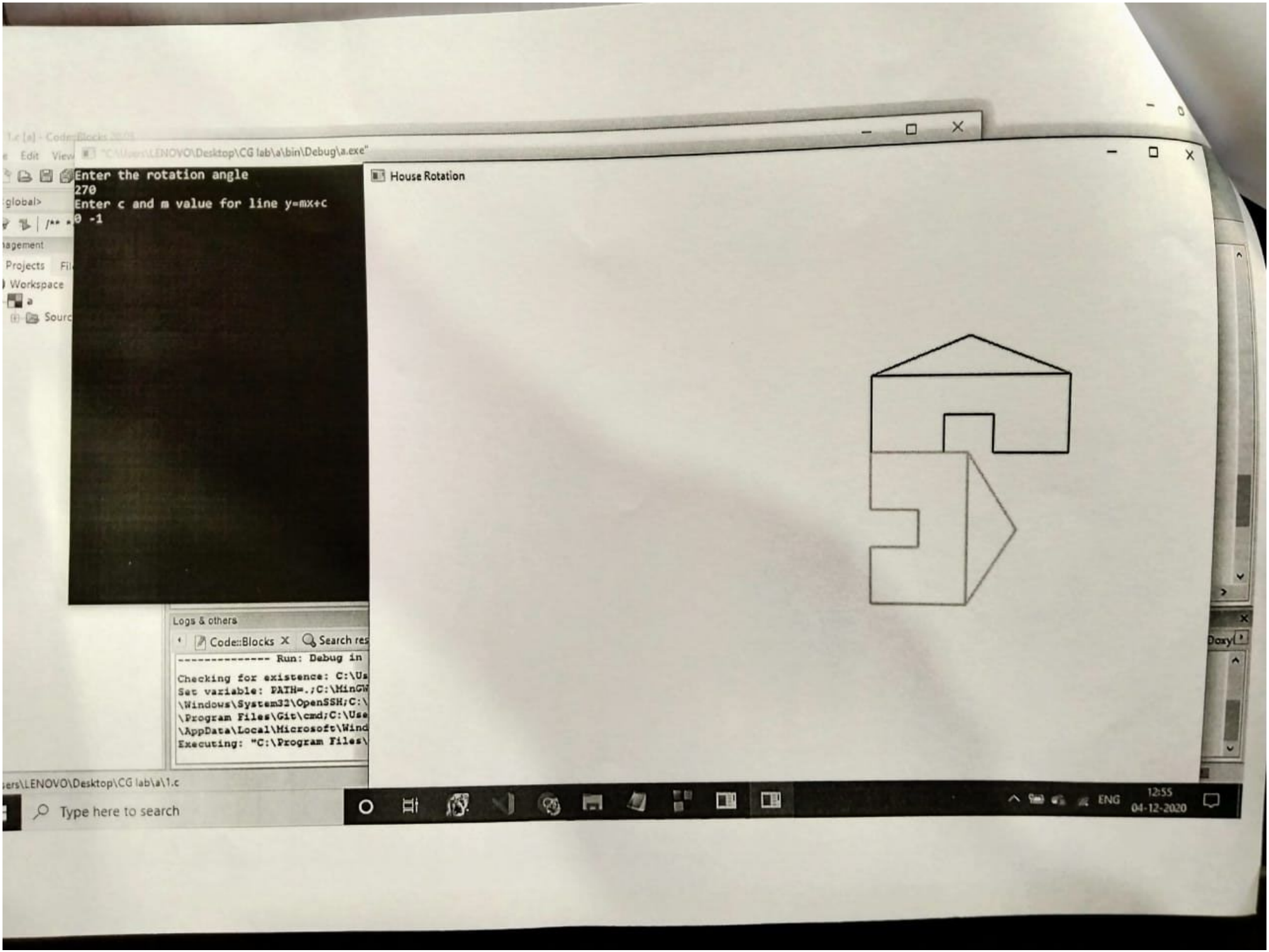
Date .....

Expt. No. 7 Page No. .....

7) Write a program for Liang Barsky algorithm

```
#include<stdio.h> #include<GL/glut.h> #include<iostream.h>
using namespace std;
double xmax, xmin, ymax, ymin, xvmax, xvmin, yvmax, yvmin;
struct lineseg { int x1, x2, y1, y2; } 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) { if(q<0) return false;
  }
  return true; }

void LiangBarsky(double x0, double x1, double y0,
                 double y1);
{ double dx = x1-x0; double dy = y1-y0;
  double u0 = 0, u2 = 1;
  if(clipTest(1,0,0);
  { glBegin(GL_LINE_LOOP);
    glVertex2f(xvmin, yvmin);
    glVertex2f(xvmax, yvmin);
    glVertex2f(xvmax, yvmax);
    glVertex2f(xvmin, yvmax);
  glEnd();
```

Teacher's Signature : _____


"C:\Users\LENOVO\Desktop\CG lab\a\bin\Debug\a.exe"
Enter the rotation angle
270
Enter c and m value for line y=mx+c
0 -1
House Rotation
Logs & others
Code::Blocks
Search res
---------- Run: Debug in
Checking for existence: C:\Us
Set variable: PATH=.;C:\MinGW
\Windows\System32\OpenSSH;C:\
\Program Files\Git\cmd;C:\Use
\AppData\Local\Microsoft\Wind
Executing: "C:\Program Files\
Type here to search
ENG
12:55
04-12-2020

```
                {
                        y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
                        x = xmin;
                }

                if (outcodeout == outcode0)
                {
                        x0 = x;
                        y0 = y;
                        outcode0 = computeoutcode(x0, y0);
                }
                else
                {
                        x1 = x;
                        y1 = y;
                        outcode1 = computeoutcode(x1, y1);
                }
        }

    } while (!done);

    if (accept)
    {
            double sx = (xvmax - xvmin) / (xmax - xmin);
            double sy = (yvmax - yvmin) / (ymax - ymin);
            double vx0 = xvmin + (x0 - xmin) * sx;
            double vy0 = yvmin + (y0 - ymin) * sy;
            double vx1 = xvmin + (x1 - xmin) * sx;
            double vy1 = yvmin + (y1 - ymin) * sy;

            glColor3f(1, 0, 0);
            glBegin(GL_LINE_LOOP);
            glVertex2f(xvmin, yvmin);
            glVertex2f(xvmax, yvmin);
            glVertex2f(xvmax, yvmax);
            glVertex2f(xvmin, yvmax);
            glEnd();

            glColor3f(0, 0, 1);
            glBegin(GL_LINES);
            glVertex2d(vx0, vy0);
            glVertex2d(vx1, vy1);
            glEnd();
    }
}

void display()
{
        double x0 = 120, y0 = 10, x1 = 40, y1 = 130;
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1, 0, 0);
        glBegin(GL_LINES);
        glVertex2d(x0, y0);
        glVertex2d(x1, y1);
        glVertex2d(60, 20);
        glVertex2d(80, 120);
        glEnd();

        glColor3f(0, 0, 1);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xmin, ymin);
        glVertex2f(xmax, ymin);
        glVertex2f(xmax, ymax);
        glVertex2f(xmin, ymax);
        glEnd();

        cohensuther(x0, y0, x1, y1);
        cohensuther(60, 20, 80, 120);
```

```
        cout << "Enter no. of lines:\n";
        cin >> n;

        for (int i = 0; i < n; i++)
        {
                cout << "Enter line endpoints (x1 y1 x2 y2):\n";
                cin >> ls[i].x1 >> ls[i].y1 >> ls[i].x2 >> ls[i].y2;

        }
        glutCreateWindow("Liang Barsky Line Clipping Algorithm");
        glutDisplayFunc(display);
        myinit();
        glutMainLoop();
}
```

Date .....

Expt. No. ..... Page No. .....

```
    LiangBarsky line ( ls[i].x1, ls[i].x2, ls[i].y1, ls[i].y2);
  glFlush();
}
void myInit(){
    glMatrixMode ( GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D ( 0, 500, 0, 500);
}

void main( int ac, char** av)
{ glutInit (&argc, av);
    glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (1000, 1000);
    glutInitWindowPos (0,0);
    cout << " Enter window co-ordinates ";
    cin >> xmin >> ymin >> xmax >> ymax;
    cout << " Enter viewport coordinates ";
    cin >> xvmin >> yvmin >> xvmax >> yvmax;
    cout << " Enter no. of lines \n";
    cin >> n;
  for (int i=0; i<n; i++)
  { cout << Enter line endpoints";
      cin >> ls[i].x1 >> ls[i].y1 >> ls[i].x2 >> ls[i].y2;
  }
  glutCreateWindow (" Liang Barsky");
  glutDisplayWindowFunc( display);
  myinit();
  glutMainLoop();
}
```

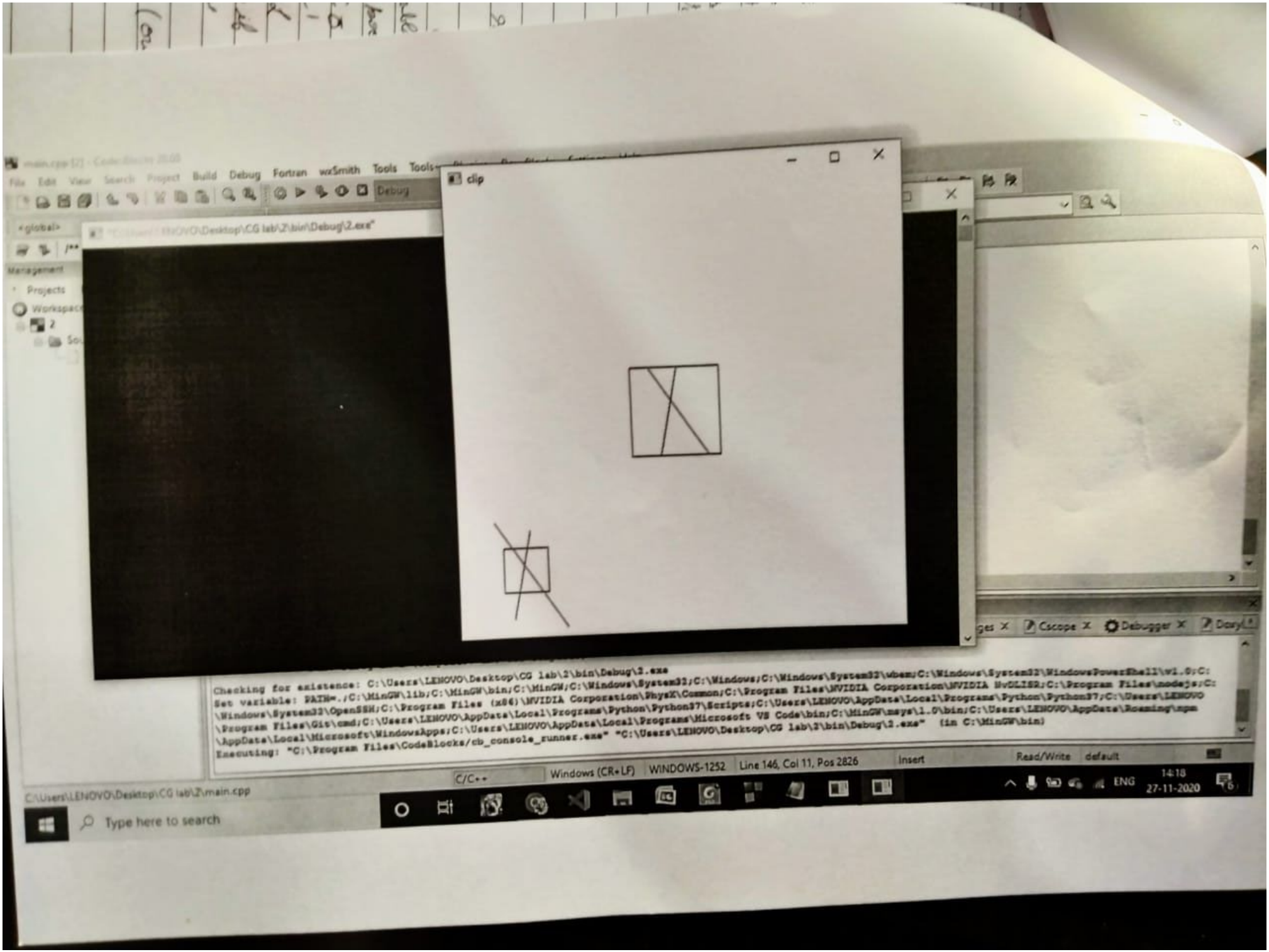
Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void clip (int poly_points [][2], int & poly_size,
           int x1, int y1, int x2, int y2)
{ int new_points [MAXPOINTS][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);
  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_pos [new_poly_size][0] = x_intersect (x1,
                  y1, x2, y2, ix, iy, Kx, Ky);
      new_points [new_poly_size][1] = y_intersect (x1, y1,
```

Teacher's Signature : _____


clip
Checking for existence: C:\Users\LENOVO\Desktop\CG lab\2\bin\Debug\2.exe
Set variable: PATH=./C:\MinGW\lib;C:\MinGW\bin;C:\MinGW;C:\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:
\Windows\System32\OpenSSH;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR;C:\Program Files\nodejs;C:
\Program Files\Git\cmd;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37\Scripts;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37;C:\Users\LENOVO
\AppData\Local\Microsoft\WindowsApps;C:\Users\LENOVO\AppData\Local\Programs\Microsoft VS Code\bin;C:\MinGW\msys\1.0\bin;C:\Users\LENOVO\AppData\Roaming\npm
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\CG lab\2\bin\Debug\2.exe" (in C:\MinGW\bin)
Cscope
Debugger
Windows (CR+LF)
WINDOWS-1252
Line 146, Col 11, Pos 2826
Insert
Read/Write
default
C/C++
ENG
14:18
27-11-2020
Type here to search

Date .....

Expt. No. ..... Page No. .....

```
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);
glPopMatrix(); glFlush(); }
void mouse (int btn, int state, int x, int y){
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    display();
if ( btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) display2();
}
void main() {  void main( int ac, char** av)
{ printf (" Enter Rotation angle"); scanf (" %d ", angl);
printf (" Enter c & m "); scanf (" %d %d", &c, &m);
glutInit (&ac, av);
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB);
glutWindowSize (500, 500);
glutInitWindowPosition ( 100, 100);
glutCreateWindow ( "Rotation");
glutDisplayFunc ( display);
glutMouseFunc ( mouse);
glutMainLoop();
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
else if (outcode1 & BOTTOM) { x = x0 + (x1 - x0)
    * (ymin - y0) / (y1 - y0); y = ymin; }
else if (outcode1 & RIGHT) { y = y0 + (y1 - y0) *
    (xmax - x0) / (x1 - x0); x = xmax; }
else { y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
    x = xmin; }
if (oc0 == oc) { x0 = x; y0 = y; oc0 =
                computecode(x0, y0); }
else { x1 = x; y1 = y; oc1 = computecode(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(); }}
```

Teacher's Signature : _____

Date .....

Expt. No. 5 .....

Page No. .....

5) Write a program to create a house
a) Reflect it about given point using OpenGL
b) reflect it about $y = mx + c$

```
#include <glGLUT.h>
#include <math.h>  #include <stdio.h>
float house[11][2] = {{100,200},{200,250},
    {300,200},{100,200},{100,100},{175,150}
{225,250},{300,100},{300,200}};
int angle; float m, c, theta;
void display() { glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION); glLoadIdentity();
glOrtho2D(-500,500,-500,500); glColor3f(1,0,0);
  glBegin(LINE_LOOP)
    for(i=0; i<11; i++) glVertex2f(house[i])
glEnd(); glFlush(); glPushMatrix();
glTranslate2f(100,100,0); glRotate2f(angle,0,0,1);
glTranslate3f(-100,-100,0); glColor3f(1,1,0);
glBegin(GL_LINE_LOOP); for(i=0; i<11; i++)
  glVertex2fv(house[i]); glEnd();
  glPopMatrix(); glFlush(); }
void display2()
{ glClearColor(1,1,1,0); glClear(GL_COLOR_BUFFER_BIT);
glMatrix(GL_PROJECTION); glLoadIdentity();
glOrtho2D(-450,450,-450,450); 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;
```

Teacher's Signature : _____

Date .....

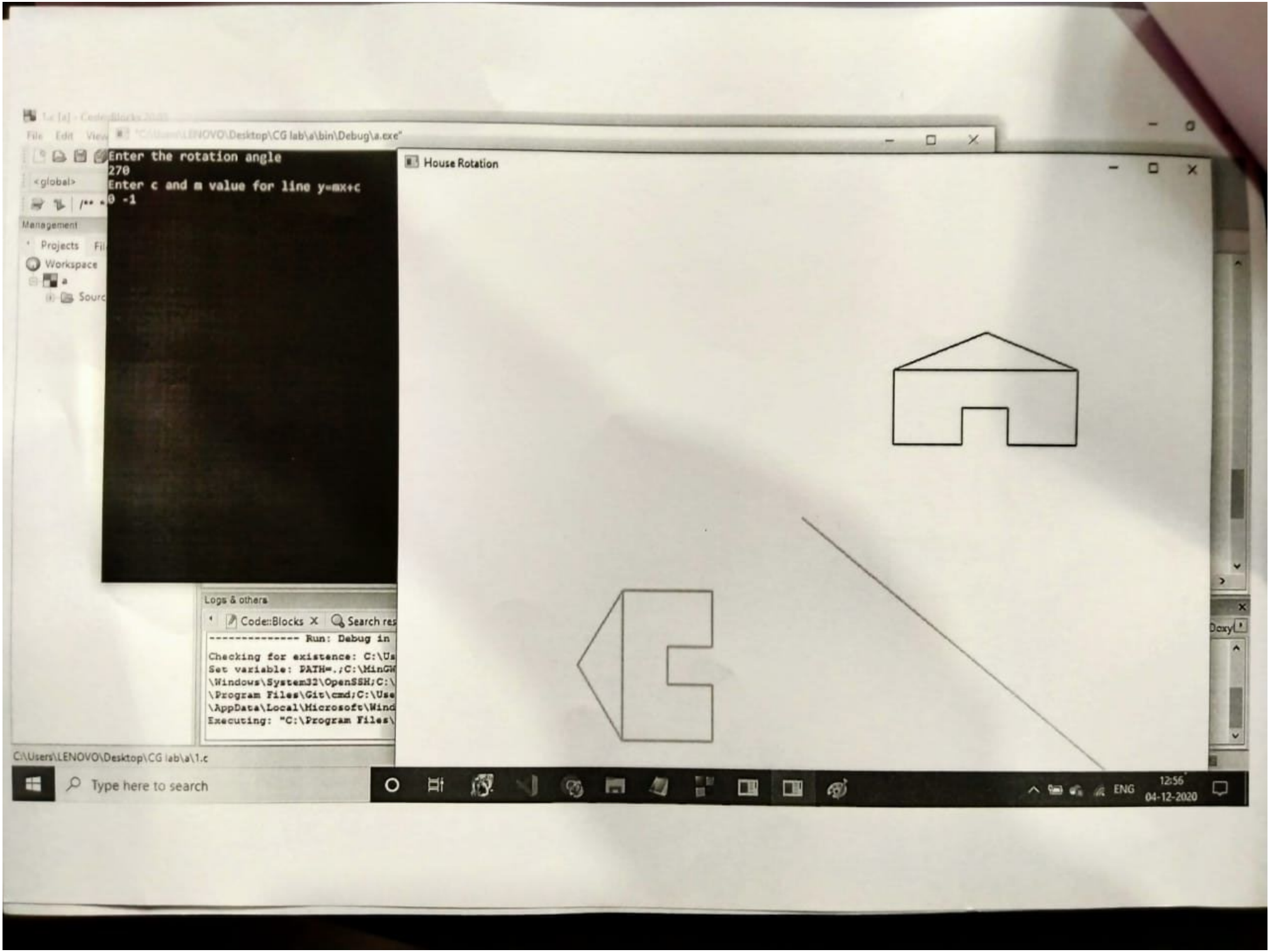
Expt. No. 8 Page No. .....

8) Write a program to implement Cohen Hodgman algorithm.

```
#include <iostream.h> #include <GL/glut.h>
using namespace std; int polygon, polygonPoints[20][2],
clip_poly_no, clip_polyPoints[20][2],
const int maxPoints = 20;

void drawPoly (int p[][2], int n)
{ glBegin (GL_POLYGON);
  for(int i=0; i<n; i++)
      glVertex2i (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) * (x3*y4 - y3*y4);
  int den = (x1-x2) * (y3-y4) - (y1-y2)
            * (x3-x4);
  return num/den;
}
```

Teacher's Signature : _____

```
                                                    y0 = y0 + u1 * dy;
                                                }
                                                // Window to viewport mappings
                                                double sx = (xvmax - xvmin) / (xmax -
xmin); // Scale parameters
                                                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); // draw blue
colored clipped line

                                                glBegin(GL_LINES);
                                                glVertex2d(vx0, vy0);
                                                glVertex2d(vx1, vy1);
                                                glEnd();
                                        }
}// end of line clipping

void display()
{
        glClear(GL_COLOR_BUFFER_BIT);
        //draw the line with red color
        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();
        }
        //draw a blue colored window
        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);

        cout << "Enter window coordinates (xmin ymin xmax ymax): \n";
        cin >> xmin >> ymin >> xmax >> ymax;
        cout << "Enter viewport coordinates (xvmin yvmin xvmax yvmax) : \n";
        cin >> xvmin >> yvmin >> xvmax >> yvmax;
```


"C:\Users\LENOVO\Desktop\CG lab\a\bin\Debug\a.exe"
Enter the rotation angle
270
Enter c and m value for line y=mx+c
0 -1
House Rotation
Logs & others
Code::Blocks
Search res
---------- Run: Debug in
Checking for existence: C:\Us
Set variable: PATH=.;C:\MinGW
\Windows\System32\OpenSSH;C:\
\Program Files\Git\cmd;C:\Use
\AppData\Local\Microsoft\Wind
Executing: "C:\Program Files\
C:\Users\LENOVO\Desktop\CG lab\a\1.c
Type here to search
ENG
12:56
04-12-2020

Date _____

Expt. No. _____ Page No. _____

```
{
    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 : %d");
        scanf ("%d", &poly_points[i][0],
                     &poly_points[i][1]);
        org_poly_points[i][0] = poly_points[i][0];
        org_poly_points[i][1] = poly_points[i][1];
    }
    printf ("Enter no. of vertices of clipping window");
    scanf ("%d", &clipper_size);
    for (int i=0; i<clipper_size; i++)
    {   scanf ("%d %d", &clippers[i][0], &clipperpoints
                                         [i][1]);
    }
    glut Init (&cc, &av);
    glut Init DisplayMode (GLUT_SINGLE | GLUT_RGB);
    glut Init WindowPos (400, 400);
    glut Window Position (100, 100);
    glut Create Window ("Polygon Clipping");
    glut Display Func (Display);
    glut Main Loop();
}
```

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)
{
        //window 1
        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); /* Enable hidden--surface--removal */
        glEnableClientState(GL_COLOR_ARRAY);
        glEnableClientState(GL_NORMAL_ARRAY);
        glEnableClientState(GL_VERTEX_ARRAY);
        glVertexPointer(3, GL_FLOAT, 0, vertices);
        glColorPointer(3, GL_FLOAT, 0, colors);
        glNormalPointer(GL_FLOAT, 0, normals);
        glColor3f(1.0, 1.0, 1.0);
        glutMainLoop();
}
```

```
#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, 0.0,0.0,1.0,
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)
{
        /* display callback, clear frame buffer and z buffer,
           rotate cube and draw, swap buffers */

        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()
{
        /* Idle callback, spin cube 2 degrees about selected axis */
        //sleep(50);
        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)
{
        /* mouse callback, selects an axis about which to rotate */
        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;
```

Date .....

Expt. No. 11 Page No. .....

11 Menu with Curves

```
#include <gl/glut.h> #include <math.h> #include
<stdio.h> struct screenPt { int x, int y; };
int curve = 1, red = 0, green = 0, blue = 0;
typedef enum { limacon = 1, cardioid = 2, threeleaf = 3,
spiral = 4 } curveName; int w = 600; int h = 500;
- void myinit() { glClearColor(1,1,1,1);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0,200,0,150); }
- void lineSegment(screenPt p1, screenPt p2) {
glBegin(GL_LINES);
glVertex2i(p1.x, p1.y); glVertex2i(p2.x, p2.y);
glEnd(); glFlush(); }
- void drawCurve(int curveNum) {
const double twoPi = 6.28185;
const int a = 175, b = 60; float r, theta;
int x0 = 200, y0 = 250; screenPt curvePt[2];
curve = curveNum;
glColor3f(red, green, blue);
curvePt[0].x = x0; curvePt[0].y = y0;
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;
} curvePt[1].x = x0 + r * cos(theta);
curvePt[1].y = y0 + r * sin(theta);
```

Teacher's Signature : _____

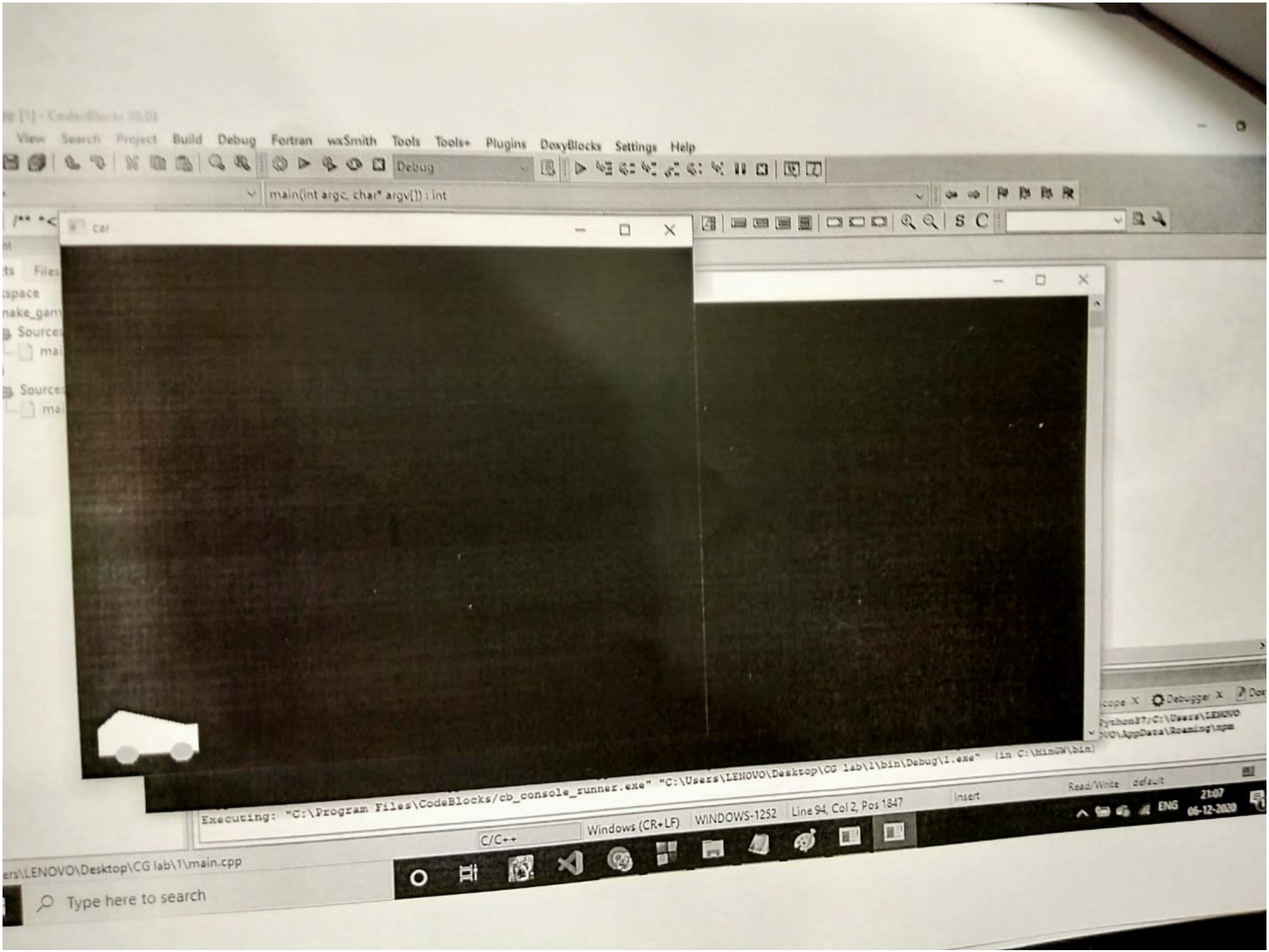
```
                green = 0;
                blue = 1;

                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() {
        /*int curveNum=1;
        glClear(GL_COLOR_BUFFER_BIT);
        /*printf("Enter the integer value corresponding to one of the followinf
curve names:\n");
        printf("1 - limacon\n2 - cardioid\n3 - threeleaf\n4 - spiral\n");
        scanf_s("%d", &curveNum);*/

        /*if (curveNum == 1 || curveNum == 2 || curveNum == 3 || curveNum == 4)
        drawCurve(curveNum);*/

}

void myreshape(int nw, int nh) {
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, (double)nw, 0.0, (double)nh);
```

```
}
int main(int argc, char* argv[]) {
        glutInit(&argc, argv);


        //USE KEYBOARD
        cout << "Enter the x co-ordinates";
        cin >> x1[0] >> x1[1] >> x1[2] >> x1[3];
        cout << "Enter y co-ordinates";
        cin >> yc[0] >> yc[1] >> yc[2] >> yc[3];
        //END KEYBOARD

        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("BZ");
        glutDisplayFunc(display);
        //glutMouseFunc(mymouse); //INCLUDE FOR MOUSE, REMOVE FOR KEYBOARD
        myInit();
        glutMainLoop();
}
```

```
}
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();
}
```

```
#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(screenPt p1, screenPt p2) {

        glBegin(GL_LINES);
        glVertex2i(p1.x, p1.y);
        glVertex2i(p2.x, p2.y);
        glEnd();
        glFlush();
}
void drawCurve(int curveNum) {
        const double twoPi = 6.283185;
        const int a = 175, b = 60;
        float r, theta, dtheta = 1.0 / float(a);
        int x0 = 200, y0 = 250;
        screenPt curvePt[2];
        curve = curveNum;
        glColor3f(red, green, blue);
        curvePt[0].x = x0;
        curvePt[0].y = y0;
        glClear(GL_COLOR_BUFFER_BIT);
        switch (curveNum) {
        case limacon: curvePt[0].x += a + b; break;
        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;
                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;
        }
}

void colorMenu(int id) {
        switch (id) {

        case 0:
                break;
        case 1:
                red = 0;
```

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

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

        glClearColor(1, 1, 1, 1);
        glColor3f(1, 1, 1);
        glPointSize(5);
        gluOrtho2D(0, 500, 0, 500);
}

void drawPixel(float x, float y) {
        glBegin(GL_POINTS);
        glVertex2f(x, y);
        glEnd();
}

void display() {

        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double t;
        glColor3f(0, 0, 0);
        glBegin(GL_POINTS);
        for (t = 0; t < 1; t = t + 0.005) {
                double xt = pow(1 - t, 3) * x1[0] + 3 * t * pow(1 - t, 2) * x1[1]
+ 3 * pow(t, 2) * (1 - t) * x1[2] + pow(t, 3) * x1[3];
                double yt = pow(1 - t, 3) * yc[0] + 3 * t * pow(1 - t, 2) * 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;


        }
```


View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
main(int argc, char* argv[]) : int
car
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\CG lab\1\bin\Debug\1.exe" (in C:\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 94, Col 2, Pos 1847
Insert
Read/Write
default
ENG
21:07
06-12-2020
Type here to search

Date

Expt. No. 12 Page No.

P) Write a program to construct Bezier curve.

```
#include<iostream.h>  #include<math.h>
#include<GL/glut.h>  using namespace std;
float b, g, r, x1[4], yc[4]; int flag=0;
void myInit() { glClearColor(1,1,1,1);
  glColor3f(1,1,1); glPointSize(5);
  gluOrtho2D(0,500,0,500); }
void drawPixel( float x, float y {
    glBegin(GL_POINTS);
      glVertex2f(x,y);
      glEnd();
}
void display() { glClear( GL_COLOUR_BUFFER_BIT);
 int i; double t;
glBegin(GL_POINTS) for(int (t=0; t<1; t+=0.005) {
double xt= pow(1-t,3) * x1[0] +3*t*pow
(1-t, 2) * x1[1] *+3 * pow(t,2) * (t-1)*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();
} }
```

Teacher's Signature :

Date .....

Expt. No. ..... Page No. .....

```
void spincube() { delay(0.01); theta[axis] += 2.0;
if (theta[axis] > 360.0) theta[axis] -= 360.0;
glutPostRedisplay();
}
void mouse (int btn, int state, int x, int y) {
if ( btn == GLUT_LEFT_BUTTON && state ==
    GLUT_DOWN) axis=0;
if ( btn == GLUT_MIDDLE_BUTTON && state
    == GLUT_DOWN) axis=1;
if ( btn == GLUT_RIGHT_BUTTON && state ==
    GLUT_DOWN) axis = 2;
}

void main (int argc, char ** argv)
{ glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition (100, 100);
  glutInitWindowSize (500, 500);
  glutCreateWindow ("colorcube");
  glutDisplayFunc( glutSingle);
                   display
  glutIdleFunc ( spincube); glutMouseFunc (mouse);
  glEnable ( GL_DEPTH_TEST);
  glEnableClientState ( GL_COLOR_ARRAY);
  glEnableClientState ( GL_NORMAL_ARRAY);
  glEnableClientState ( GL_VERTEX_ARRAY);
  glColor3f (1, 1, 1);
  glutMainLoop();
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void colorMenu(int id){ switch(id){ case 0: break;
case 1: red=0; blue=1, green=0; break;
case 2: red=0; green=1, blue=0; break;
case 3: red=1, green=0; blue=0; break; default: break; }
void main_menu(int id){ switch(id){ switch(id){
case 3: exit(0); default: break(); } }
void mydisplay(){ glClear(GL_COLOR_BUFFER_BIT);
printf("1-Limacon, 2-Cardioid, 3-Threefold, 4-Normal");
scanf("%d", &curveNum); drawCurve(curveNum); }
void main(int argc, char** argv){
glutInit(&argc, argv); glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(w,h); glutInitWindowPosition(100,100);
glutCreateWindow("Curves"); int cu = glutCreateMenu(drawCurve);
glutAddMenuEntry("Limacon", 1); glutAddMenuEntry("Cardioid", 2);
glutAddMenuEntry("Threefold", 3); glutAddMenuEntry("Spiral", 4);
int colorID = glutCreateMenu(colorMenu);
glutAddMenuEntry("Red", 4); glutAddMenuEntry("Green", 2);
glutAddMenuEntry("Blue", 1); glutAddMenuEntry("Black", 0);
glutAttachMenu(GLUT_LEFT_BUTTON);
glutCreateMenu(main_menu);
glutAddSubMenu("drawCurve", cu);
glutAddSubMenu("colors", colorID);
glutAddMenuEntry("quit", 3);
glutAttachMenu(GLUT_LEFT_BUTTON);
myInit();
glutDisplayFunc(mydisplay);
glutMainLoop();
}
```

Teacher's Signature : _____

```
        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("Black", 0);
        glutAddMenuEntry("Yellow", 6);
        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();
}
```

Date .....

Expt. No. ..... Page No. .....

```
void mymouse (int btn, int state, int x, int y)
{ if (btn == GLUT_LEFT_BUTTON && state ==
      GLUT_DOWN )
  { x1[flag] = x;
    yc[flag] = 500 - y; cout << x << y;
    glPointSize(3); glColor3f (1,1,0); glBegin(GL_POINTS);
    glVertex2i( x, 500 - y); glEnd(); glFlush();
    flag++;
  }
}
int main (int argc, char** argv[]) {
  glutInit (&argc, argv);
  cout << "Enter x cordinates";
  cin >> x1[0] >> x1[1] >> x1[2] >> x1[3];
  cout << "Enter y coordinates";
  cin >> y1[0] >> y1[1] >> y1[2] >> y1[3];
  glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB);
  glutInitWindowPosition ( 100, 100);
  glutInitWindowSize (500, 500);
  glutCreateWindow("B2");
  glutDisplayFunc (display); myInit();
  glutMainLoop();
}
```

Teacher's Signature : _____

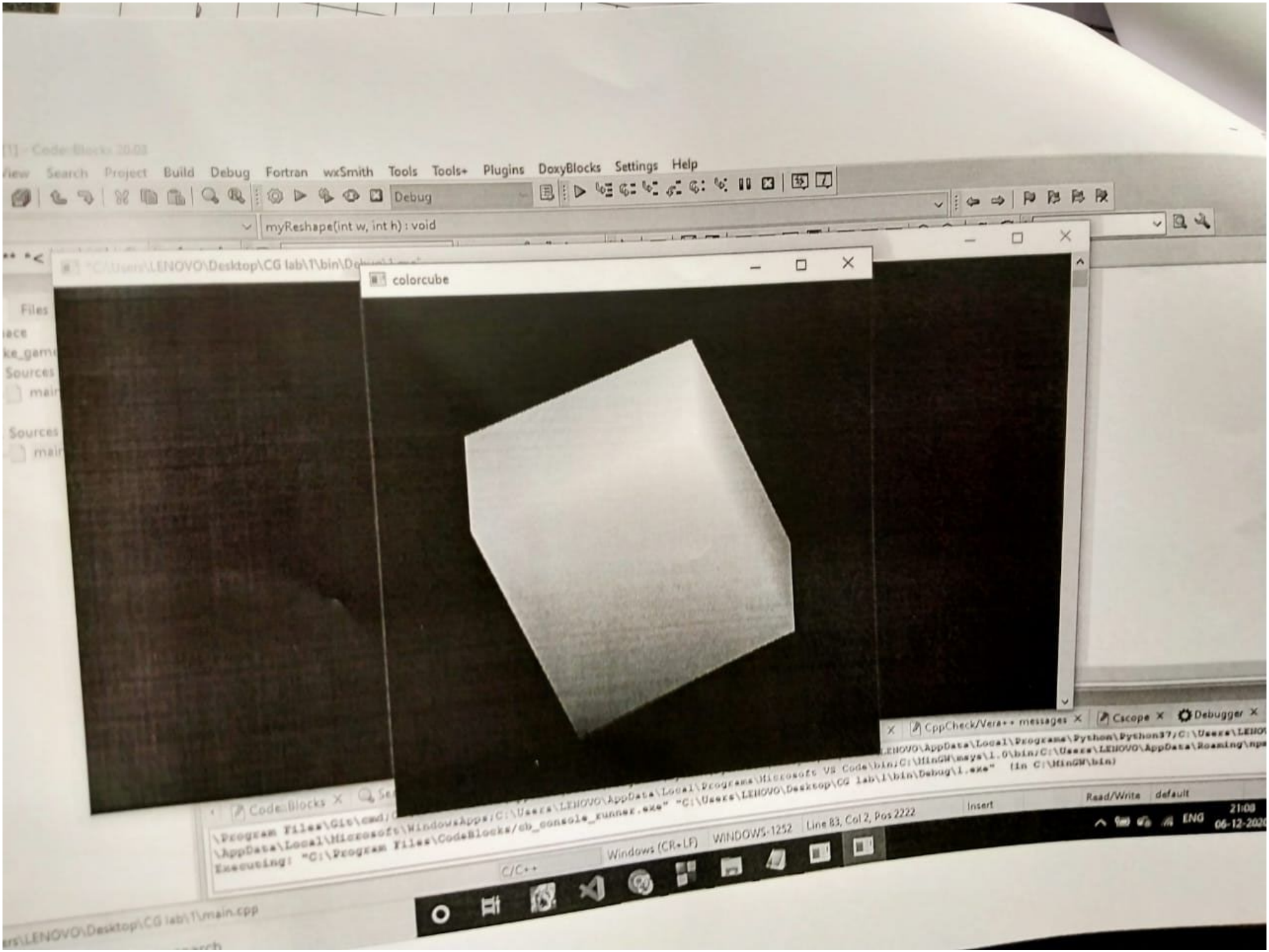
Date .....

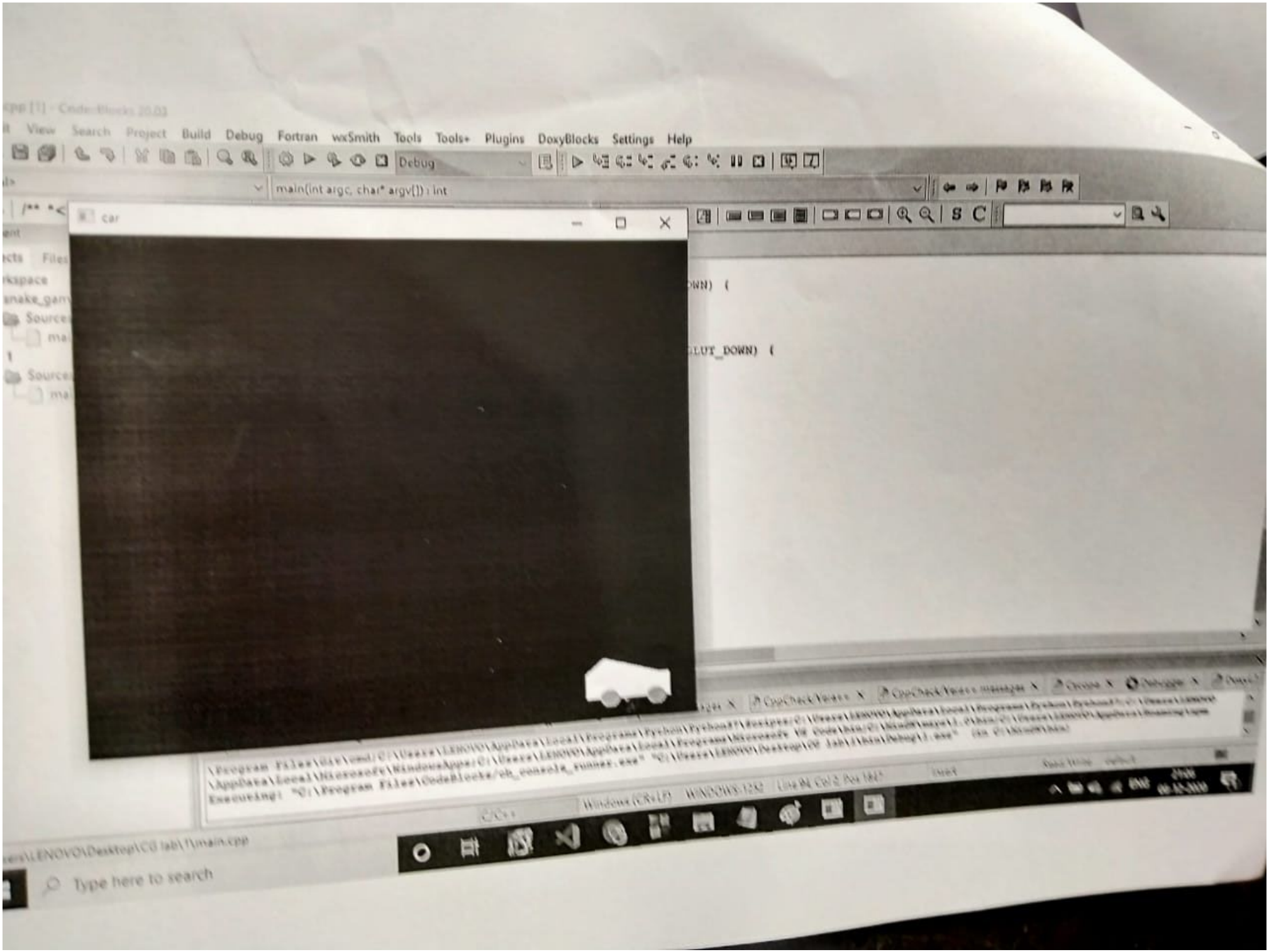
Expt. No. Model a car using display list Page No. .....

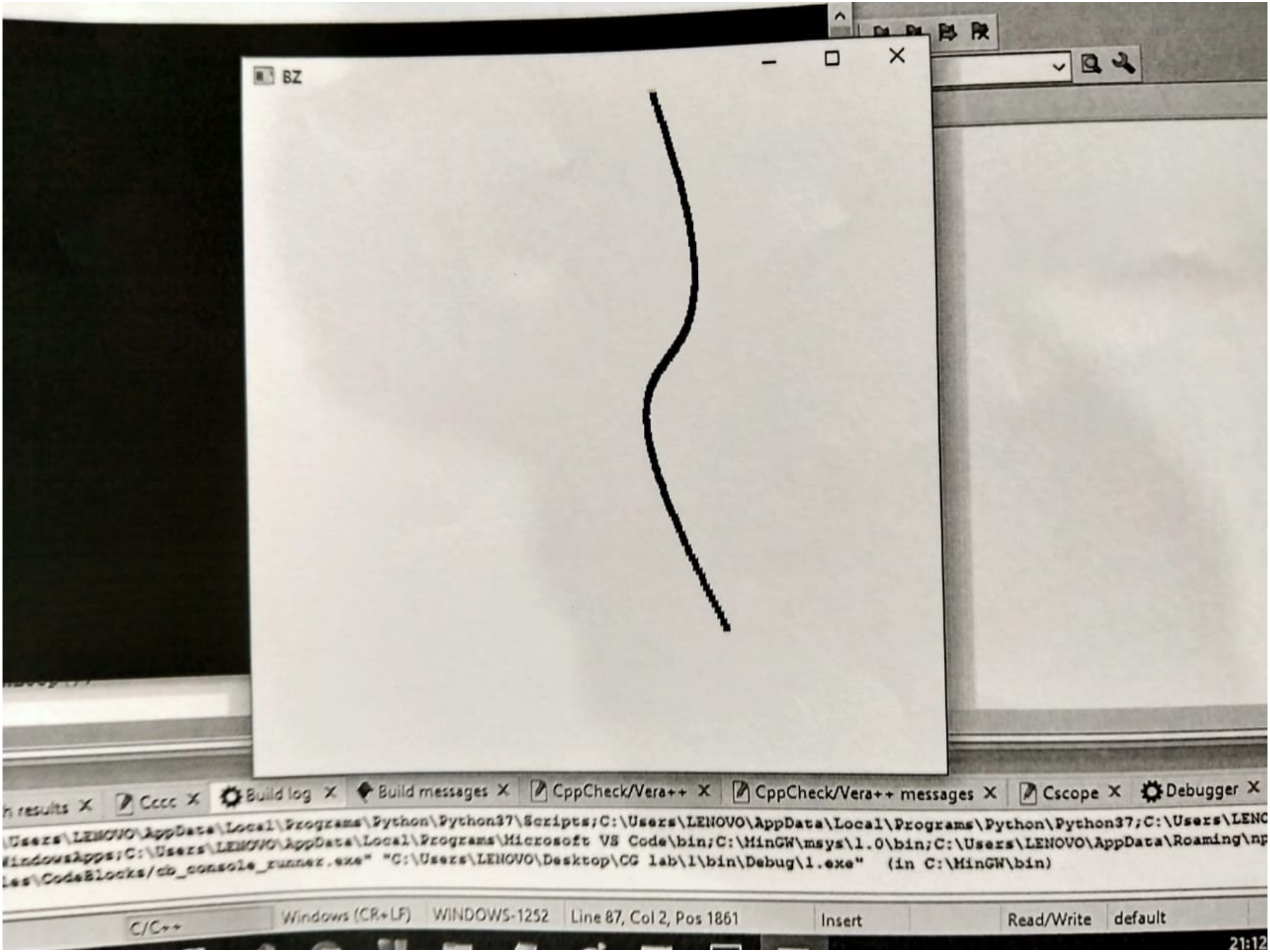
a] Write a program to model car using list move it from one end to another control speed with mouse

```
#include <GL/glut.h> #include<math.h> #include<stdlib.h>
#define AXLE  #define WHEEL 2
float s=1;
void carlist(){ glNewList(GL_COMPILE);
  glColor3f(1,1,1);
  glBegin(GL_POLYGON);
      glVertex3f(0,25,0);
      glVertex3f(90,25,0);
      glVertex3f(90,35,0);
      glVertex3f(80,55,0);
      glVertex3f(20,75,0);
      glVertex3f(0,55,0);
  glEnd(); glEndList(); }

void wheellist(){ glNewList(WHEEL, GL_COMPILE,AND_EVOLVE
  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);
}
```

Teacher's Signature : _____


colorcube
Debug
myReshape(int w, int h) : void


View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
main(int argc, char* argv[]) : int
car
Type here to search

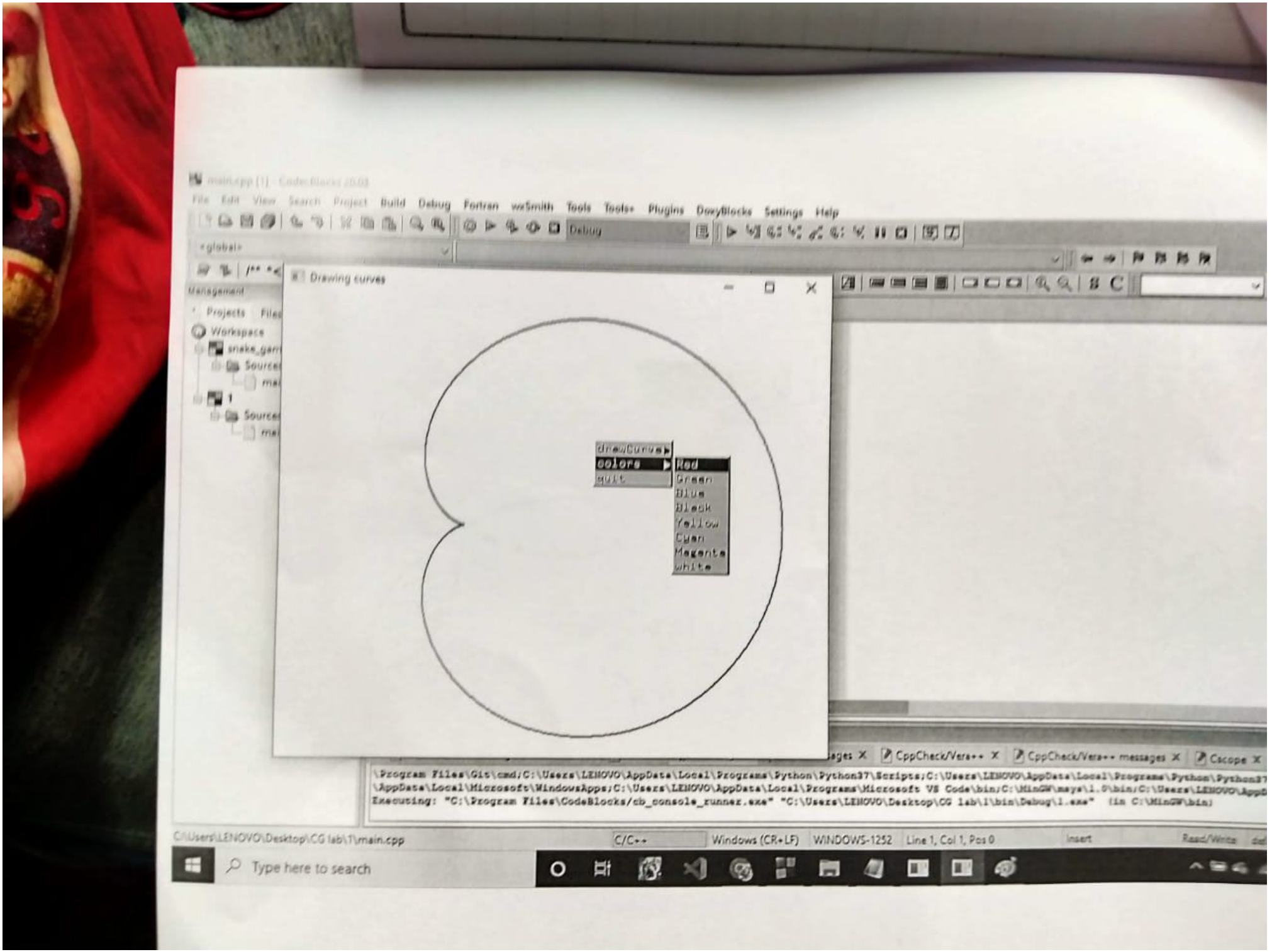

BZ
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
C:\Users\LENOVO\AppData\Local\Programs\Python\Python37\Scripts;C:\Users\LENOVO\AppData\Local\Programs\Python\Python37;C:\Users\LENO
C:\Users\LENOVO\AppData\Local\Programs\Microsoft VS Code\bin;C:\MinGW\msys\1.0\bin;C:\Users\LENOVO\AppData\Roaming\np
"C:\Users\LENOVO\Desktop\CG lab\1\bin\Debug\1.exe" (in C:\MinGW\bin)
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 87, Col 2, Pos 1861
Insert
Read/Write
default

Date .....

Expt. No. ..... Page No. .....

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

void moveCar(float s){ glTranslate(s,0,0);
   glCallList(CAR);
   glPushMatrix(); glTranslate(25,25,0);
   glCallList(WHEEL);
   glPopMatrix(); glPushMatrix();
   glTranslatef(75,25,0); glCallList(WHEEL);
   glFlush();
}
void mydisp() { glClear(GL_COLOR_BUFFER_BIT);
  carlist(); moveCar(s); wheelList(); }

                              int state
void mouse ~~()~~ (int btn, int x, int y){
   if (btn == GLUT_LEFT_BUTTON && GLOT_DOWN)
      { st=5; myDisp(); }
   if (btn == GLUT_RIGHT_BUTTON && state == GLUT_
      DOWN) { st=2; myDisp(); }
}
int main(int ac, char** av[]) {
glutInit(&ac, &av); glutInitDisplayMode(GLUT_RGB);
glutInitWindowSize(500,500); glutInitWindowPos(100,100);
glutCreateWindow("Car"); myInit();
glutDisplayFunc(myDisp);
glutMouseFunc(mouse);
glutKeyboardFunc(myKeyboard); glutMainLoop();
}
```

Teacher's Signature : _____


Drawing curves
drawCurves
colors
quit
Red
Green
Blue
Black
Yellow
Cyan
Magenta
White
Type here to search

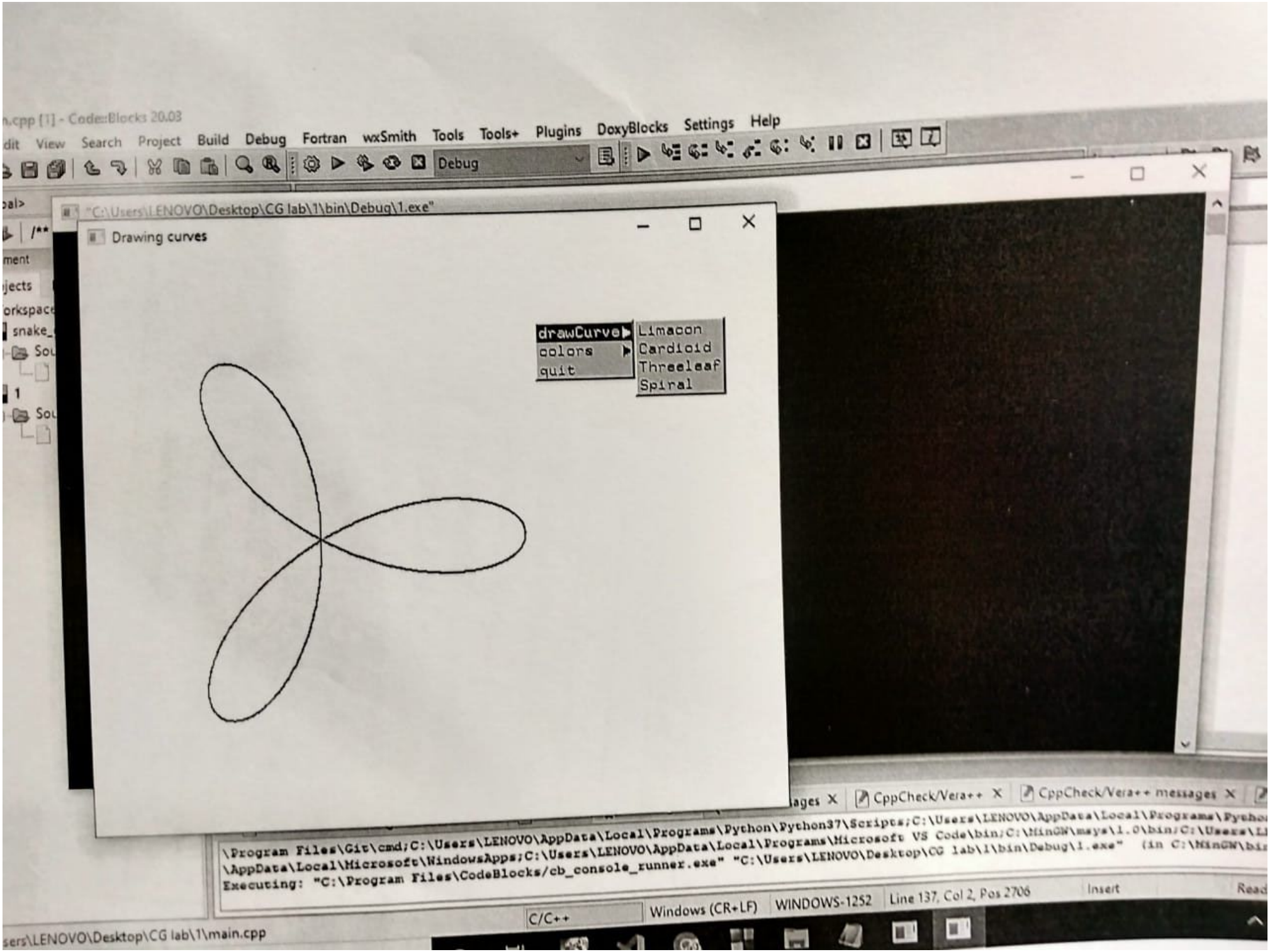
Date .....

Expt. No. Colour Cube (10) Page No. .....

10) Create a Colour Cube and spin it using OpenGL

```
#include<stdio.h> #include<GL/glut.h> #include<time.h>
GLfloat vertices[] = { -1,-1,-1, 1,-1,-1, 1,1,-1, -1,1,-1,
-1,-1,1, 1,-1,1, 1,1,1, -1,1,1 };
GLfloat normals[] = { -1,-1,-1, 1,-1,-1, 1,1,-1, -1,1,-1,
-1,-1,1, 1,-1,1, 1,1,1, -1,1,1 };
GLfloat colors[] = { 0,0,0, 1,0,0, 1,1,0, 0,0,1,
0,0,0, 1,0,0, 1,1,0, 0,1,1 };
GLubyte cubeIndices[] = { 0,3,2,1, 3,7,6,0, 4,7,3,2,
2,6,5,4, 5,6,7,0, 1,5,4,0 };
GLfloat theta[] = {0.0,0.0,0.0};
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)
glLoadIdentity();
glRotatef(theta[0],1,0,0);
glRotatef(theta[1],0,1,0);
glRotatef(theta[2],0,0,1);
glDrawElements(GL_QUADS, GL_UNSIGNED_BYTE,
                cubeIndices);
glBegin(GL_LINES);
    glVertex3f(0,0,0); glVertex3f(1,1,1);
glEnd(); glFlush();
}
```

Teacher's Signature : _____


Drawing curves
drawCurve
colors
quit
Limacon
Cardioid
Threeleaf
Spiral
Debug
Executing: "C:\Program Files\CodeBlocks/cb_console_runner.exe" "C:\Users\LENOVO\Desktop\CG lab\1\bin\Debug\1.exe"
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 137, Col 2, Pos 2706
Insert