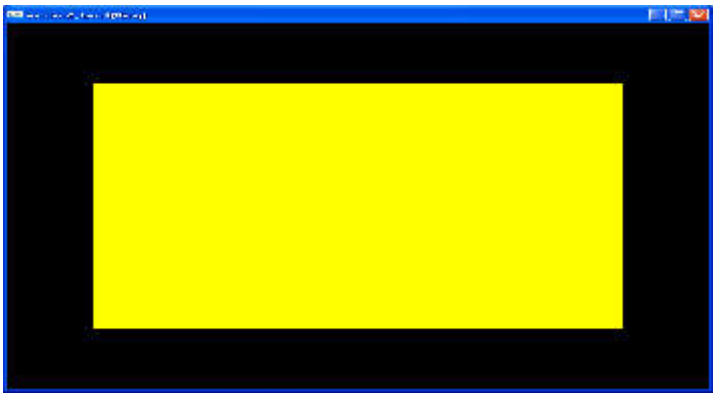
----Session 2----

Exercise 4:WAP to display yellow rectangle in openGl.

```
#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>

void init()
{
      glClearColor(0.0,0.0,0.0,0.0); // background Color
      glColor3f(1.0,1.0,0.0);//Drawing Color
      glPointSize(4);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,800,0.0,600.0);
}
void display()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_POLYGON);
            glVertex2i(100,100);
            glVertex2i(100,500);
            glVertex2i(700,500);
            glVertex2i(700,100);
      glEnd();
      glFlush();
}
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutCreateWindow("Session2, Exer4(Niraj)");
      glutDisplayFunc(display);
      init();
      glutMainLoop();
```

}

Output:



Exercise 5:

```
/*
 * Problem definition: Yello rectange on black background
 * By: Dr. A. K. Marandi
 * Date: 01/01/2019
 */
#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>

void init()
{
	glClearColor(0.0,0.0,0.0,0.0); // background Color
	glColor3f(1.0,0.0,0.0);//Drawing Color
	glPointSize(4);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,800,0.0,600.0);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
```

```
        glBegin(GL_POLYGON);
                glVertex2i(100,100);
                glVertex2i(100,500);
                glVertex2i(700,500);
                glVertex2i(700,100);
        glEnd();
        glFlush();
}
void main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(800,600);
        glutCreateWindow("Session2, Exer4(Niraj)");
        glutDisplayFunc(display);
        init();
        glutMainLoop();
}
```

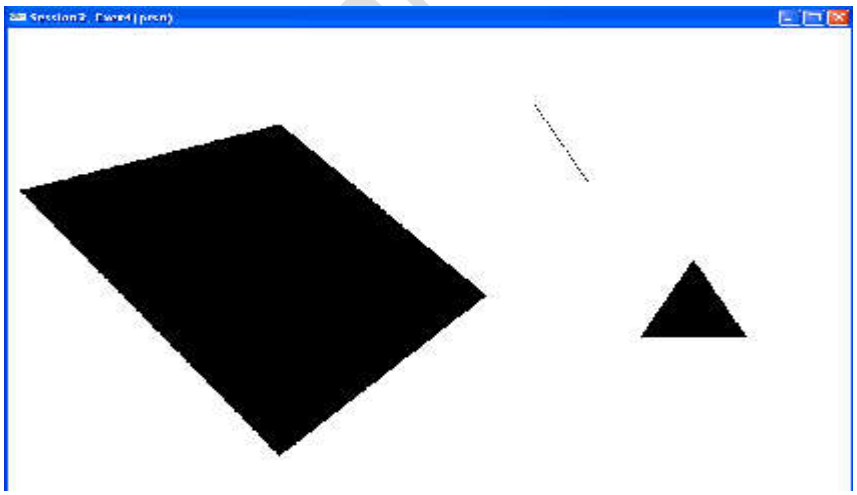
Output:

Exercise 6:

```
/*
 * Problem definition: just to draw basic primitives of OpenGl
 * such as GL_INES, GL_QUAD, GL_TRIANGLES etc.
 * By: Dr. A. K. Marandi
 * Date: 01/01/2019
 */

#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>

void init()
{
      glClearColor(1.0,1.0,1.0,0.0); //background Color
      glColor3f(0.0,0.0,0.0);//Drawing Color
      glPointSize(4);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,800,0.0,600.0);
}
void display()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_LINES);
            glVertex2i(500,500);
            glVertex2i(550,400);
      glEnd();

      glBegin(GL_QUADS);
            glVertex2i(125,350);
            glVertex2i(362,254);
            glVertex2i(154,352);
            glVertex2i(346,254);
            glVertex2i(258,45);
            glVertex2i(454,253);
            glVertex2i(259,475);
```

```
            glVertex2i(12,389);
      glEnd();

      glBegin(GL_TRIANGLES);
            glVertex2i(600,200);
            glVertex2i(650,300);
            glVertex2i(700,200);
      glEnd();

      glFlush();
}
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutCreateWindow("Session2, Exer4(prsn)");
      glutDisplayFunc(display);
      init();
      glutMainLoop();
}
```

Output:



----Session 3----

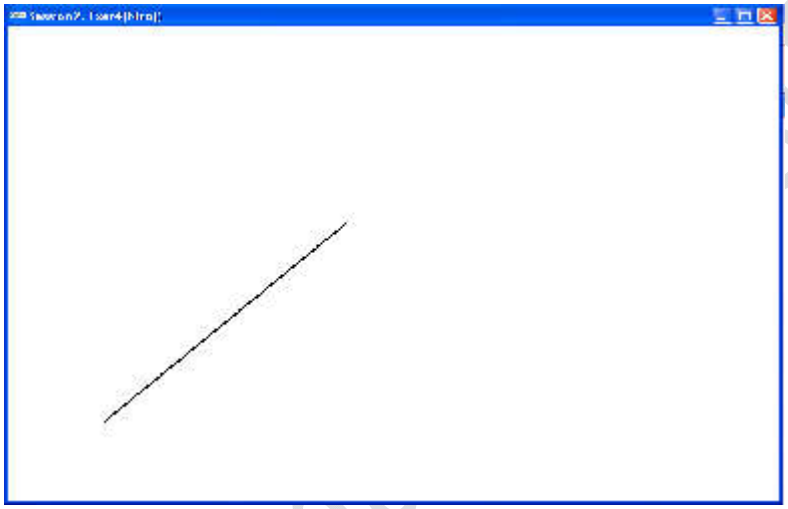
Exercise 7:

```
/*
 * Problem definition:To impement DDA Line-generation algorithm
 * By: Dr. A. K. Marandi
 * Date: 01/01/2019
 */

#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>
#include<math.h>
void nkjSwap(float *,float *);

void nkjInit()
{
      glClearColor(1.0,1.0,1.0,0.0); //Black background Color
      glColor3f(0.0,0.0,0.0);//Drawing Color yellow
      glPointSize(2);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,800,0.0,600.0);
}

void nkjDDA4f(float x1, float y1, float x2, float y2)
{
    if(x1>x2)
      {
            nkjSwap(&x1,&x2);
            nkjSwap(&y1,&y2);
      }

      float slope=(y2-y1)/(float)(x2-x1);
      if(slope>0 && slope<1)
      {
```

```
            while(x1<=x2)
            {
                  glVertex2i((int)x1,int(y1));
                  x1++;
                  y1+=slope;
            }
      }
      else if(slope>=1)
      {
            float slope1=1/slope;
            while(y1<=y2)
            {
                  glVertex2i(int(x1),int(y1));
                  y1++;
                  x1+=slope1;
            }

      }
}

void nkjSwap(float *x, float *y)
{
      float temp=*x;
      *x=*y;
      *y=temp;
}

void nkjDisplay()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_POINTS);
            nkjDDA4f(100.0,100.0,350.0,350.0);//two points
p1(100,100), p2(350,350)
      glEnd();
      glFlush();
}
```

```
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutCreateWindow("Session2, Exer4(Niraj)");
      glutDisplayFunc(nkjDisplay);
      nkjInit();
      glutMainLoop();

}
```

Output:



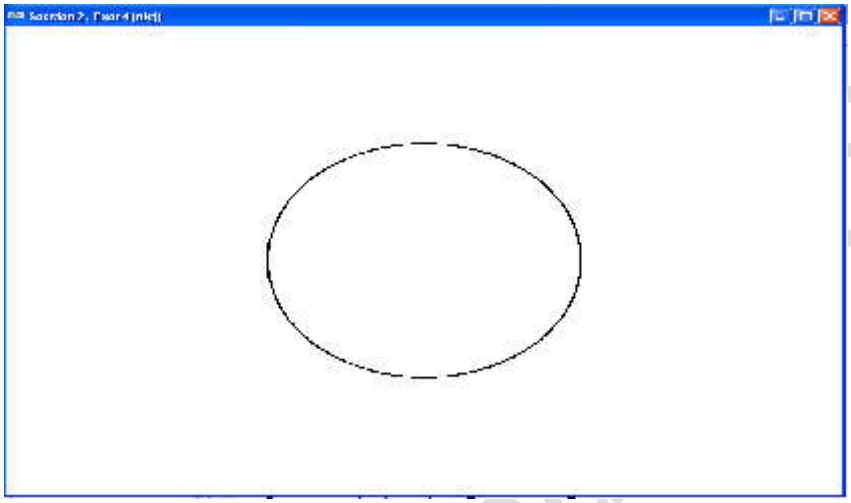
Exercise 8:

```
/*
 * Problem definition:To impement Bresenham Circle-generation
algorithm
 * By: Dr. A. K. Marandi
 * Date: 01/01/2019
 */

#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>
#include<math.h>
#include<stdio.h>

void nkjSwap(float *,float *);

void nkjInit()
{
	glClearColor(1.0,1.0,1.0,0.0); //Black background Color
	glColor3f(0.0,0.0,0.0);//Drawing Color yellow
	glPointSize(2);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,800,0.0,600.0);
}

void nkjBresenhamCircleGeneration3f(float x1, float y1, float
radious)
{
	float dcsnPrmtr=5/4-radious;//decision Parameter
	int k=0;
	float x=x1, y=y1;
	x1=0;
	y1=radious;
	while(x1<y1)
	{
		if(dcsnPrmtr<0)
```

```
            {
                  dcsnPrmtr += 2 * x1 + 2 + 1;
                  x1++;
            }
            else //if(dcsnPrmtr
            {
                  dcsnPrmtr += 2 * x1 + 2 - 2 * y1 - 2 + 1;
                  x1++;
                  y1--;
            }
            //generate symmetry points

            glVertex2i((int)(x+x1),(int)(y+y1));
            glVertex2i((int)(x-x1),(int)(y+y1));
            glVertex2i((int)(x+x1),(int)(y-y1));
            glVertex2i((int)(x-x1),(int)(y-y1));

            glVertex2i((int)(x+y1),(int)(y+x1));
            glVertex2i((int)(x-y1),(int)(y+x1));
            glVertex2i((int)(x+y1),(int)(y-x1));
            glVertex2i((int)(x-y1),(int)(y-x1));
      }
}
void nkjDisplay()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_POINTS);
            nkjBresenhamCircleGeneration3f(400.0,300.0,150.0);
      glEnd();
      glFlush();
}
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutCreateWindow("Session2, Exer4(nkj)");
      glutDisplayFunc(nkjDisplay);
```

```
	nkjInit();
	glutMainLoop();
}
```

Output:



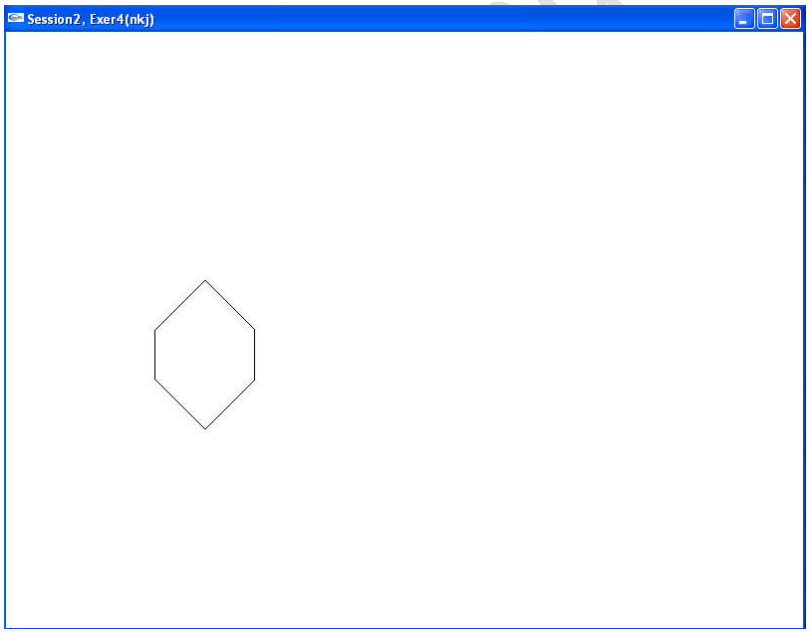
Exercise 9:

```
/*
 * Problem definition:To draw a convex hall polygon
 * By: Dr. A. K. Marandi
 * Date: 01/01/2019
 */

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

```
void prsnInit()
{
	glClearColor(1.0,1.0,1.0,0.0); //Black background Color
	glColor3f(0.0,0.0,0.0);//Drawing Color yellow
	glPointSize(4);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,800,0.0,600.0);
}

void prsnConvexHallPolygon()
{
	glVertex2i(200,200);
	glVertex2i(150,250);

	glVertex2i(150,250);
	glVertex2i(150,300);

	glVertex2i(150,300);
	glVertex2i(200,350);

	glVertex2i(200,350);
	glVertex2i(250,300);

	glVertex2i(250,300);
	glVertex2i(250,250);

	glVertex2i(250,250);
	glVertex2i(200,200);

}


void prsnDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_LINES);
```

```
            prsnConvexHallPolygon();
      glEnd();
      glFlush();
}


void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutCreateWindow("Session2, Exer4(prsn)");
      glutDisplayFunc(prsnDisplay);
      prsnInit();
      glutMainLoop();

}
```

Output:



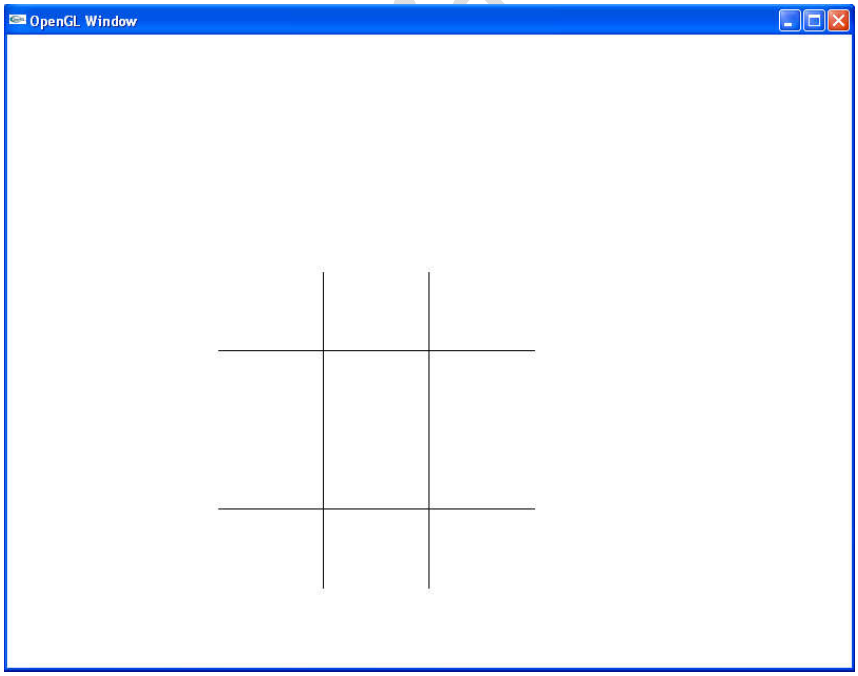
----Session 4----

Exercise 10:

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

void display();
void myInit();
void main(int i,char **ptr)
{
      glutInit(&i, ptr);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutInitWindowPosition(100,100);
      glutCreateWindow("OpenGL Window");
      glutDisplayFunc(display);
     myInit();
      glutMainLoop();
}
void myInit()
{
      glClearColor(1.0,1.0,1.0,0.0);
      glColor3f(0.0f,0.0f,0.0f);
      glPointSize(4.0);
      glMatrixMode(GL_PROJECTION);
```

```
	glLoadIdentity();
	gluOrtho2D(0.0,800.0,0.0,800.0);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_LINES);

	glVertex2i(200,200);
	glVertex2i(500,200);
	glVertex2i(200,400);
	glVertex2i(500,400);
	glVertex2i(300,100);
	glVertex2i(300,500);
	glVertex2i(400,500);
	glVertex2i(400,100);

	glEnd();
	glFlush();
}
```

Output:

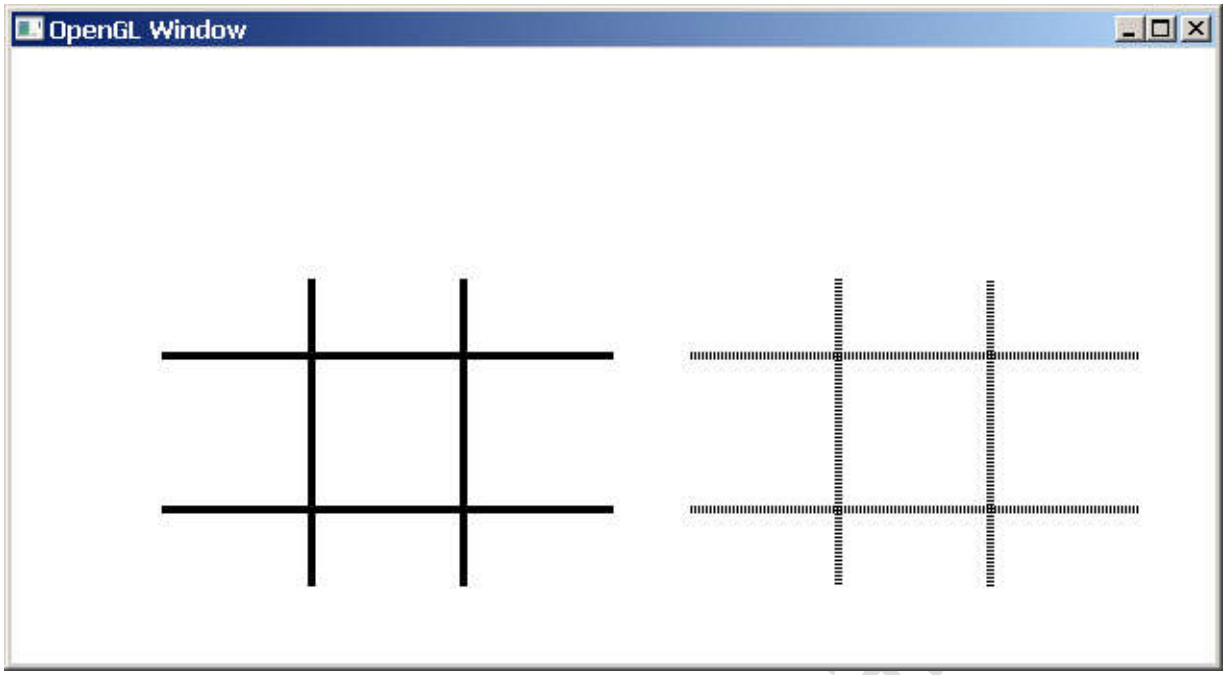


Exercise 11:

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

void display();
void myInit();
void main(int i,char **ptr)
{
      glutInit(&i, ptr);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(800,600);
      glutInitWindowPosition(100,100);
      glutCreateWindow("OpenGL Window");
      glutDisplayFunc(display);
      myInit();
      glutMainLoop();
}

void myInit()
{
      glClearColor(1.0,1.0,1.0,0.0);
      glColor3f(0.0f,0.0f,0.0f);
      glPointSize(4.0);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,800.0,0.0,800.0);
}
void display()
{
      glClear(GL_COLOR_BUFFER_BIT);
      glLineWidth(4.0);
      glBegin(GL_LINES);
            glVertex2i(100,200);
            glVertex2i(400,200);
```

```
            glVertex2i(100,400);
            glVertex2i(400,400);
            glVertex2i(200,100);
            glVertex2i(200,500);
            glVertex2i(300,500);
            glVertex2i(300,100);
      glEnd();

      glEnable(GL_LINE_STIPPLE);
      glLineStipple(1,0xAAAA);
      glBegin(GL_LINES);

      glVertex2i(450,200);
      glVertex2i(750,200);
      glVertex2i(450,400);
      glVertex2i(750,400);
      glVertex2i(550,100);
      glVertex2i(550,500);
      glVertex2i(650,500);
      glVertex2i(650,100);

      glEnd();
      glDisable(GL_LINE_STIPPLE);

      glFlush();
}
```

Output:



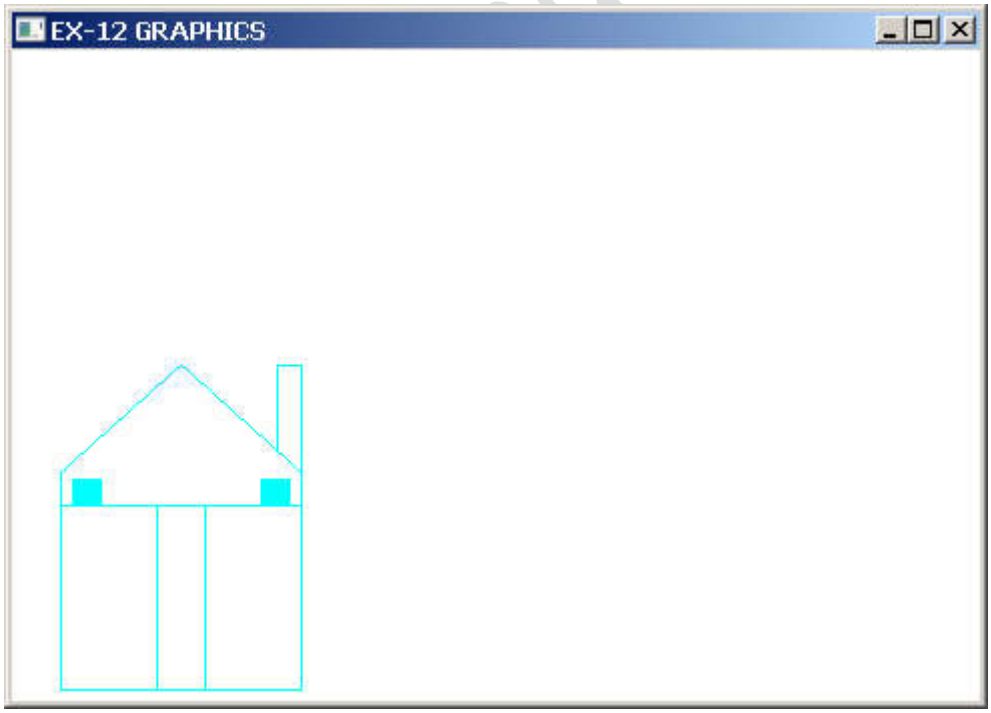
Exercise 12:

```
#include<windows.h>
#include<gl/gl.h>
#include<gl/glu.h>
#include<gl/glut.h>
void display();

void main(int argc, char *argv[])
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowPosition(50,50);
      glutCreateWindow("EX-12 GRAPHICS");
      glutDisplayFunc(display);

      //init
      glClearColor(1.0,1.0,1.0,0.0);
      glColor3f(1.0,1.0,0.0);
      glPointSize(4);
```

```
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,800.0,0.0,600.0);
	glutMainLoop();
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0.0,1.0,1.0);
	glBegin(GL_LINES);
	glVertex2i(140,310);
	glVertex2i(40,210);
	glVertex2i(140,310);
	glVertex2i(240,210);
	glVertex2i(40,210);
	glVertex2i(40,10);
	glVertex2i(240,210);
	glVertex2i(240,10);
	glVertex2i(40,180);
	glVertex2i(240,180);
	glVertex2i(240,210);
	glVertex2i(240,310);
	glVertex2i(240,310);
	glVertex2i(220,310);
	glVertex2i(220,310);
	glVertex2i(220,230);
	glVertex2i(120,180);
	glVertex2i(120,10);
	glVertex2i(120,10);
	glVertex2i(160,10);
	glVertex2i(160,10);
	glVertex2i(160,180);
	glVertex2i(40,10);
	glVertex2i(240,10);
	glEnd();


	glBegin(GL_QUADS);
```

```
        glVertex2i(50,205);
        glVertex2i(50,180);
        glVertex2i(75,180);
        glVertex2i(75,205);
        glVertex2i(205,205);
        glVertex2i(205,180);
        glVertex2i(230,180);
        glVertex2i(230,205);
        glEnd();
        glFlush();
}
```

Output:



----Session 5----

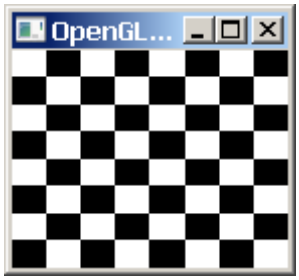
Exercise 13:

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

void display();
void myInit();
void main(int i,char **ptr)
{
      glutInit(&i, ptr);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
      glutInitWindowSize(400,400);
      glutInitWindowPosition(100,100);
      glutCreateWindow("OpenGL Window");
```

```
	glutDisplayFunc(display);
	myInit();
	glutMainLoop();
}
void myInit()
{
	glClearColor(1.0,1.0,1.0,0.0);
	glColor3f(0.0f,0.0f,0.0f);
	glPointSize(4.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,800.0,0.0,800.0);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	int i=0,j=0;
	int scale=100;
	for(i=0;i<8;i++)
	{
		for(j=0;j<8;j++)
		{
			if((i+j)%2==0)
			{
				glColor3f(0.0,0.0,0.0);
			}
			else
			{
				glColor3f(1.0,1.0,1.0);
			}
			glBegin(GL_POLYGON);
				glVertex2i(j*scale,i*scale);
				glVertex2i(j*scale,i*scale+scale);
				glVertex2i(j*scale+scale,i*scale+scale);
				glVertex2i(j*scale+scale,i*scale);
			glEnd();
		}
	}
```

```
        glFlush();
}
```

Output:



Exercise 15:

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

//function that implements Sutherand-Cohen algorithm
void nkjImpementsSutherlandCohen(int [], int , ... );
//function to deside visibiity of any line
int nkjDecideVisibility(int [],int *,int *,int *,int *);
//function to generate bit code of points
int nkjGenerateCode(int,int, int, int, int ,int);
//to perform swapping
void nkjSwap(int * , int *);

void nkjInit()
```

```
{
      glClearColor(1.0,1.0,1.0,0.0);
      glColor3f(0.0f,0.0f,0.0f);
      glPointSize(4);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,200.0,0.0,200.0);
}

void nkjDisplayLines()
{
      int points[]={110,110,10,10};// points for window position
      int xMax,yMax,xMin,yMin;
      xMax=yMax=110;
      xMin=yMin=10;
      glClear(GL_COLOR_BUFFER_BIT);

      //Drawing Window
      glBegin(GL_LINES);
            glVertex2i(xMin,yMin);
            glVertex2i(xMin,yMax);

            glVertex2i(xMin,yMax);
            glVertex2i(xMax,yMax);

            glVertex2i(xMax,yMax);
            glVertex2i(xMax,yMin);

            glVertex2i(xMax,yMin);
            glVertex2i(xMin,yMin);


      nkjImpementsSutherlandCohen(points,12,25,75,90,100,125,35,110,1
20,50,50,120,120);
      glEnd();
      glFlush();
}
void nkjImpementsSutherlandCohen(int polygonPoints[], int
vertexPoints, ... )
```

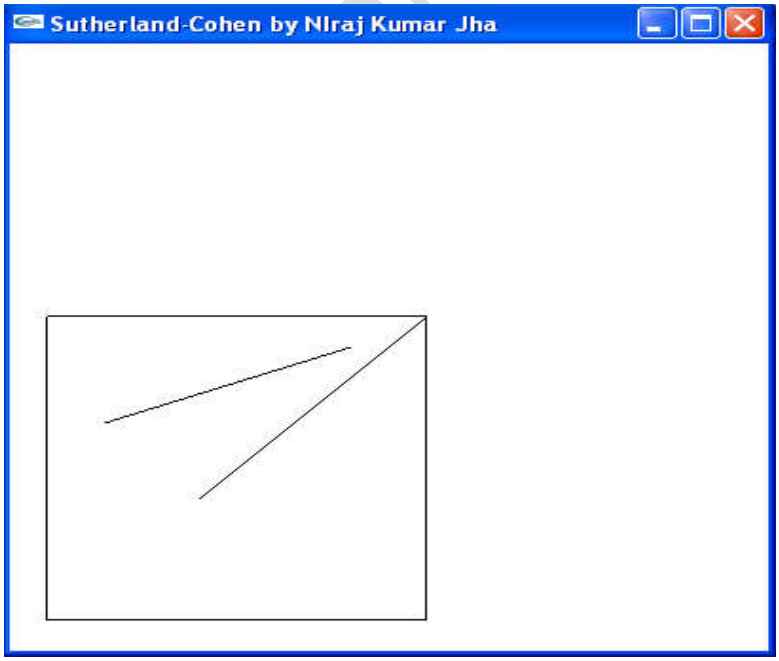
```
{
      int x1, y1, x2,y2;
      int ind, total, decision;

      va_list ptr;
      va_start(ptr, vertexPoints);
      if(vertexPoints%4!=0)
      {
            printf("nkjError Message! Wrong number of arguments
given.....\n");
            return;
      }
      total=vertexPoints/4;
      glClear(GL_COLOR_BUFFER_BIT);

      for(ind=0;ind<total;ind++)
      {
            x1=va_arg(ptr,int);
            y1=va_arg(ptr,int);
            x2=va_arg(ptr,int);
            y2=va_arg(ptr,int);
            decision=
nkjDecideVisibility(polygonPoints,&x1,&y1,&x2,&y2);
            if(decision!=-1)
            {
            //this implies ine must be drawn and points are stored
            //in the corresponding variables
                  glVertex2i(x1,y1);
                  glVertex2i(x2,y2);
            }
      }
}

int nkjDecideVisibility(int points[], int *x1,int *y1, int *x2, int
*y2)
{
      int xMax,yMax,xMin,yMin;
      int code1,code2;
      xMax=points[0];
```

```
yMax=points[1];
xMin=points[2];
yMin=points[3];

for(;;)
{
      code1=nkjGenerateCode(xMax,yMax,xMin,yMin,*x1,*y1);
      code2=nkjGenerateCode(xMax,yMax,xMin,yMin,*x2,*y2);

      if(code1==0 && code2==0)
      {
            //this indicates line is totaly visible
            return 1;
      }
      else if((code1 & code2)!=0)
      {
            //this implies line is totaly invisible
            return -1;
      }
      else
      {
            if(*x1>xMax)
            {
      //finding intersection of line[(x1,y1),(x2,y2)] and xMax
                  *y1=(((*y2-*y1)/(*x2-*x1))*(xMax-*x1)) + *y1;
                  *x1=xMax;
            }
            else if(*x1<xMin)
            {
      //finding intersection of line[(x1,y1),(x2,y2)] and xMin
                  *y1=(((*y2-*y1)/(*x2-*x1))*(xMin-*x1)) + *y1;
                  *x1=xMin;
            }

            if(*y1>yMax)
            {
      //finding intersection of line[(x1,y1),(x2,y2)] and yMax
                  *x1=((yMax-*y1)*((*x2-*x1)/(*y2-*y1))) + *x1;
```

```
                        *y1=yMax;
                }
                else if(*y1<yMin)
                {
        //finding intersection of line[(x1,y1),(x2,y2)] and yMin
                        *x1=((yMin-*y1)*((*x2-*x1)/(*y2-*y1))) + *x1;
                        *y1=yMin;
                }
        }

        //generating new code for the clipped points

        code1=nkjGenerateCode(xMax,yMax,xMin,yMin,*x1,*y1);
        if(code1==0)
        {
                //interchange two points and respective flags
                nkjSwap(x1,x2);
                nkjSwap(y1,y2);
                nkjSwap(&code1,&code2);
        }
    }
    return -1; //this will never execute, just to satisfy compiler
}

int nkjGenerateCode(int xMax, int yMax, int xMin, int yMin, int x,
int y)
{
    int code=0;
    //code sequence UDLR
    if(x>xMax)
        code|=1;//0001 Right bit
    else if(x<xMin)
        code|=2;//0010 Left bit

    if(y>yMax)
        code|=8;//1000 Up/Top bit
    else if(y<yMin)
        code|=4;//0100 Down/Bottom nit
```

```
	return code;
}

void nkjSwap(int *x, int *y)
{
	*x=*x^*y;
	*y=*x^*y;
	*x=*x^*y;
}

void main(int argc, char **argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(400,400);
	glutInitWindowPosition(10,10);
	glutCreateWindow("Sutherland-Cohen by Prasun Pal");
	glutDisplayFunc(nkjDisplayLines);
	nkjInit();
	glutMainLoop();
}
```

Output:



----Session 6----

Exercise 16:

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

//function that implements Sutherand-Cohen algorithm
void nkjImpementsSutherlandCohen(int [], int , ... );
//function to deside visibiity of any line
int nkjDecideVisibility(int [],int *,int *,int *,int *);
//function to generate bit code of points
int nkjGenerateCode(int,int, int, int, int ,int);
//to perform swapping
void nkjSwap(int * , int *);

void nkjInit()
{
      glClearColor(1.0,1.0,1.0,0.0);
      glColor3f(0.0f,0.0f,0.0f);
      glPointSize(4);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
    gluOrtho2D(0.0,200.0,0.0,200.0);
}

void nkjDisplayLines()
{
int points[]={60,40,20,20};// points for window position xMax, yMax,
                           // xMin, yMin
      int xMax,yMax,xMin,yMin;
      xMax=60;
```

```
	yMax=40;
	xMin=yMin=20;
	glClear(GL_COLOR_BUFFER_BIT);

	//Drawing Window
	glBegin(GL_LINES);
		glVertex2i(xMin,yMin);
		glVertex2i(xMin,yMax);

		glVertex2i(xMin,yMax);
		glVertex2i(xMax,yMax);

		glVertex2i(xMax,yMax);
		glVertex2i(xMax,yMin);

		glVertex2i(xMax,yMin);
		glVertex2i(xMin,yMin);

	//Total 4 points two for p and two for q
	nkjImpementsSutherlandCohen(points,4,40,80,120,30);
	glEnd();
	glFlush();
}

void nkjImpementsSutherlandCohen(int polygonPoints[], int
vertexPoints, ... )
{
	int x1, y1, x2,y2;
	int ind, total, decision;

	va_list ptr;
	va_start(ptr, vertexPoints);
	if(vertexPoints%4!=0)
	{
printf("nkjError Message! Wrong number of arguments given......\n");
		return;
	}
	total=vertexPoints/4;
```

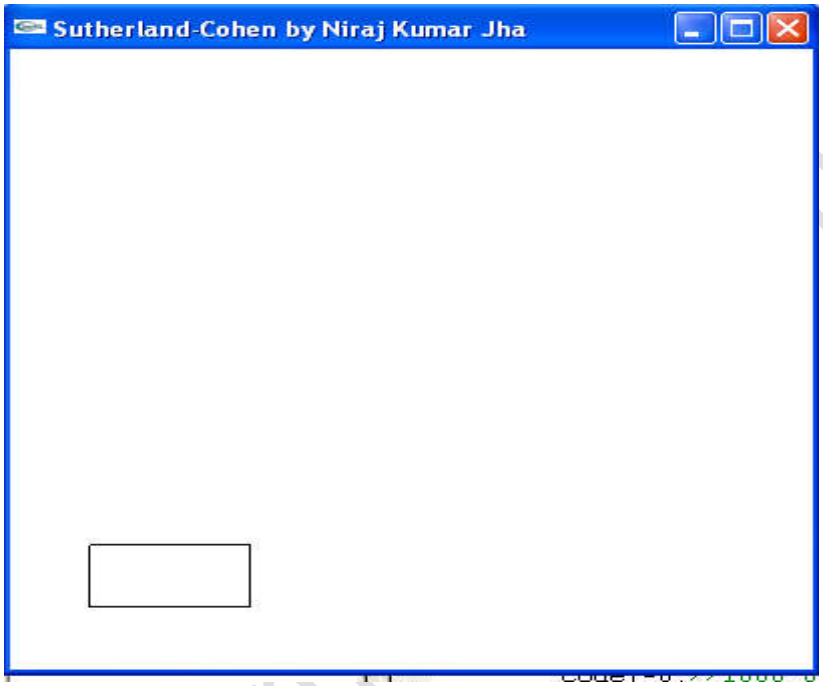
```
	glClear(GL_COLOR_BUFFER_BIT);

	for(ind=0;ind<total;ind++)
	{
		x1=va_arg(ptr,int);
		y1=va_arg(ptr,int);
		x2=va_arg(ptr,int);
		y2=va_arg(ptr,int);
		decision=
nkjDecideVisibility(polygonPoints,&x1,&y1,&x2,&y2);
		if(decision!=-1)
		{
		//this implies ine must be drawn and points are stored
		//in the corresponding variables
			glVertex2i(x1,y1);
			glVertex2i(x2,y2);
		}
	}
}

int nkjDecideVisibility(int points[], int *x1,int *y1, int *x2, int
*y2)
{
	int xMax,yMax,xMin,yMin;
	int code1,code2;
	xMax=points[0];
	yMax=points[1];
	xMin=points[2];
	yMin=points[3];

	for(;;)
	{
		code1=nkjGenerateCode(xMax,yMax,xMin,yMin,*x1,*y1);
		code2=nkjGenerateCode(xMax,yMax,xMin,yMin,*x2,*y2);

		if(code1==0 && code2==0)
		{
			//this indicates line is totaly visible
```

```
      return 1;
}
else if((code1 & code2)!=0)
{
      //this implies line is totaly invisible
      return -1;
}
else
{
      if(*x1>xMax)
      {
//finding intersection of line[(x1,y1),(x2,y2)] and xMax
            *y1=(((*y2-*y1)/(*x2-*x1))*(xMax-*x1)) + *y1;
            *x1=xMax;
      }
      else if(*x1<xMin)
      {
//finding intersection of line[(x1,y1),(x2,y2)] and xMin
            *y1=(((*y2-*y1)/(*x2-*x1))*(xMin-*x1)) + *y1;
            *x1=xMin;
      }

      if(*y1>yMax)
      {
//finding intersection of line[(x1,y1),(x2,y2)] and yMax
            *x1=((yMax-*y1)*((*x2-*x1)/(*y2-*y1))) + *x1;
            *y1=yMax;
      }
      else if(*y1<yMin)
      {
//finding intersection of line[(x1,y1),(x2,y2)] and yMin
            *x1=((yMin-*y1)*((*x2-*x1)/(*y2-*y1))) + *x1;
            *y1=yMin;
      }
}

//generating new code for the clipped points
```

```
            code1=nkjGenerateCode(xMax,yMax,xMin,yMin,*x1,*y1);
            if(code1==0)
            {
                  //interchange two points and respective flags
                  nkjSwap(x1,x2);
                  nkjSwap(y1,y2);
                  nkjSwap(&code1,&code2);
            }
      }
      return -1; //this will never execute, just to satisfy compiler
}

int nkjGenerateCode(int xMax, int yMax, int xMin, int yMin, int x,
int y)
{
      int code=0;
      //code sequence UDLR
      if(x>xMax)
            code|=1;//0001 Right bit
      else if(x<xMin)
            code|=2;//0010 Left bit

      if(y>yMax)
            code|=8;//1000 Up/Top bit
      else if(y<yMin)
            code|=4;//0100 Down/Bottom nit

      return code;
}

void nkjSwap(int *x, int *y)
{
      *x=*x^*y;
      *y=*x^*y;
      *x=*x^*y;
}

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

```
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(400,400);
	glutInitWindowPosition(10,10);
	glutCreateWindow("Sutherland-Cohen by Niraj Kumar Jha");
	glutDisplayFunc(nkjDisplayLines);
	nkjInit();
	glutMainLoop();
}
```

Output:



Output for the given window size, where line is totally invisible.