Computer Graphics and Visualization Laboratory Programs
Implement the following programs in C / C++

1. **POINTS**

```
#include<stdio.h>
#include<GL/glut.h>
void myinit()
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,500,0,500);
	glMatrixMode(GL_MODELVIEW);
}
void disp1()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1,0,0);
	glPointSize(5);
		glBegin(GL_POINTS);
			glVertex2f(25,25);
			glVertex2f(250,250);
		glEnd();
	glFlush();
}
void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Points");
	myinit();
	glutDisplayFunc(disp1);
	glutMainLoop();
}
```

2. **LINES**

```
#include<stdio.h>
#include<GL/glut.h>
void myinit()
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,100,0,100);
	glMatrixMode(GL_MODELVIEW);
}
void disp1()
{
	glClearColor(1,1,1,1);
```

```
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
        glLineWidth(10);
                glBegin(GL_LINES);
                        glVertex2f(25,25);
                        glVertex2f(75,75);
                glEnd();
        glFlush();
}
void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(300,150);
        glutCreateWindow("Lines");
        myinit();
        glutDisplayFunc(disp1);
        glutMainLoop();
}
```

3. LINELOOP

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

float v[3][2] = {{25,25},{50,25},{40,50}};

void myinit()
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,100,0,100);
        glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
                glBegin(GL_LINE_LOOP);
                        glVertex2fv(v[0]);
                        glVertex2fv(v[1]);
                        glVertex2fv(v[2]);
                glEnd();
        glFlush();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Triangle");
	myinit();
	glutDisplayFunc(disp1);
	glutMainLoop();
}
```

4. TRIANGLE

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

float v[3][2] = {{25,25},{50,25},{40,50}};

void myinit()
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,100,0,100);
	glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1,0,0);
		glBegin(GL_LINES);
			glVertex2fv(v[0]);
			glVertex2fv(v[1]);
			glVertex2fv(v[1]);
			glVertex2fv(v[2]);
			glVertex2fv(v[2]);
			glVertex2fv(v[0]);
		glEnd();
	glFlush();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Triangle");
	myinit();
	glutDisplayFunc(disp1);
```

```
        glutMainLoop();
}
```

5. RECTANGLE

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

float v[4][2] = {{25,25},{50,25},{50,50},{25,50}};

void myinit()
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,100,0,100);
        glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
                glBegin(GL_LINES);
                        glVertex2fv(v[0]);
                        glVertex2fv(v[1]);
                        glVertex2fv(v[1]);
                        glVertex2fv(v[2]);
                        glVertex2fv(v[2]);
                        glVertex2fv(v[3]);
                        glVertex2fv(v[3]);
                        glVertex2fv(v[0]);
                glEnd();
        glFlush();
}

void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(300,150);
        glutCreateWindow("Triangle");
        myinit();
        glutDisplayFunc(disp1);
        glutMainLoop();
}
```

6. COLOR TRIANGLE

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

float v[3][2] = {{25,25},{50,25},{40,50}};

void myinit()
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,100,0,100);
        glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
                glBegin(GL_LINES);
                        glVertex2fv(v[0]);
                        glVertex2fv(v[1]);
                glEnd();
        glColor3f(0,1,0);
                glBegin(GL_LINES);
                        glVertex2fv(v[1]);
                        glVertex2fv(v[2]);
                glEnd();
        glColor3f(0,0,1);
                glBegin(GL_LINES);
                        glVertex2fv(v[2]);
                        glVertex2fv(v[0]);
                glEnd();
        glFlush();
}

void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(300,150);
        glutCreateWindow("Triangle");
        myinit();
        glutDisplayFunc(disp1);
        glutMainLoop();
}
```

7. COLOR RECTANGLE

```
#include<GL/glut.h>

float v[4][2] = {{25,25},{50,25},{50,50},{25,50}};

void myinit()
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,100,0,100);
	glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1,0,0);
		glBegin(GL_POLYGON);
			glVertex2fv(v[0]);
			glVertex2fv(v[1]);
			glVertex2fv(v[2]);
			glVertex2fv(v[3]);
		glEnd();
	glFlush();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Triangle");
	myinit();
	glutDisplayFunc(disp1);
	glutMainLoop();
}
```

8. QUAD STRIP

```
#include<GL/glut.h>

float v[4][2] = {{25,25},{50,25},{25,50},{50,50}};
float u[4][2] = {{50,50},{75,50},{50,75},{75,75}};

void myinit()
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,100,0,100);
	glMatrixMode(GL_MODELVIEW);
}

void disp1()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1,0,0);
		glBegin(GL_QUAD_STRIP);
			glVertex2fv(v[0]);
			glVertex2fv(v[1]);
			glVertex2fv(v[2]);
			glVertex2fv(v[3]);
			glVertex2fv(u[0]);
			glVertex2fv(u[1]);
			glVertex2fv(u[2]);
			glVertex2fv(u[3]);
		glEnd();
	glFlush();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Triangle");
	myinit();
	glutDisplayFunc(disp1);
	glutMainLoop();
}
```

9. **STROKED CIRCLE**

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

void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(-50,50,-50,50);
	glMatrixMode(GL_MODELVIEW);
}

void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);

	int i;
	float x1,x2,y1,y2,r1=15,r2=18,t;

	glColor3f(1,0,0);
	glBegin(GL_QUAD_STRIP);
		for(i=0;i<=24;i++)
		{
			t = 3.142/12 * i;
			x1 = r1*cos(t);
			y1 = r1*sin(t);
			x2 = r2*cos(t);
			y2 = r2*sin(t);
			glVertex2f(x1,y1);
			glVertex2f(x2,y2);
		}
	glEnd();

	glFlush();
}


void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);

	glutCreateWindow("Stroked O");
	myinit();
	glutDisplayFunc(display);
	glutMainLoop();
}
```

10. 2D SIERPINSKI GASKET

```
#include<stdlib.h>
#include<GL/glut.h>
float v[3][2] = {{-25,-25},{0,25},{25,-25}};
void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(-50,50,-50,50);
	glMatrixMode(GL_MODELVIEW);
}

void display()
```

```
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
		glBegin(GL_LINE_LOOP);
			glVertex2fv(v[0]);
			glVertex2fv(v[1]);
			glVertex2fv(v[2]);
		glEnd();
	float p[2] = {0,0};
	int i,n=5000,j;
	glPointSize(2);

	for(i=0;i<n;i++)
	{
		j=rand()%3;
		if(j==0)
			glColor3f(1,0,0);
		else if(j==1)
			glColor3f(0,1,0);
		else
			glColor3f(0,0,1);
		p[0] = (p[0] + v[j][0])/2;
		p[1] = (p[1] + v[j][1])/2;
		glBegin(GL_POINTS);
			glVertex2fv(p);
		glEnd();
		glFlush();
	}
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);
	glutCreateWindow("2D Sierpinkski Gasket");
	myinit();
	glutDisplayFunc(display);
	glutMainLoop();
}
```

11. 2D_Recursive_Sierpinski

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

void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(-50,50,-50,50);
	glMatrixMode(GL_MODELVIEW);
}

void disp_tri(float v0[2],float v1[2],float v2[2])
{
	glBegin(GL_POLYGON);
		glVertex2fv(v0);
		glVertex2fv(v1);
		glVertex2fv(v2);
	glEnd();
}


void div_tri(float v0[2],float v1[2],float v2[2],int n)
{
	float a[2],b[2],c[2];
	int i;

	if(n>0)
	{
		for(i=0;i<2;i++)
		{
			a[i] = (v0[i] + v1[i])/2;
			b[i] = (v1[i] + v2[i])/2;
			c[i] = (v0[i] + v2[i])/2;
		}
		div_tri(v0,a,c,n-1);
		div_tri(a,v1,b,n-1);
		div_tri(b,c,v2,n-1);
	}
	else
		disp_tri(v0,v1,v2);
}

int c;

void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);


	float v[3][2] = {{-25,-25},{0,25},{25,-25}};

	char str[]="2D Sierpinski Gasket";
	int i;
```

```
        glColor3f(0,0,1);
        glRasterPos2i(-20,40);
        for(i=0;i<strlen(str);i++)
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
        glColor3f(1,0,0);
        div_tri(v[0],v[1],v[2],c);
        glFlush();
}

void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(300,150);
        printf("Enter the number of subdivisions:\n");
        scanf("%d",&c);
        glutCreateWindow("2D Recursive Sierpinski");
        myinit();
        glutDisplayFunc(display);
        glutMainLoop();
}
```

12. 3D_Recursive_Sierpinski

Program to recursively subdivide a tetrahedron to from 3D Sierpinski gasket. The number of recursive steps is to be specified by the user.

```
#include<stdio.h>
#include<gl/glut.h>
int c;
float v[4][3] = {{0,0,1},{-1,-0.5,0},{0,1,0},{1,-0.5,0}};
void myinit()
{
        glMatrixMode(GL_PROJECTION_MATRIX);
        glLoadIdentity();
        glOrtho(-2,2,-2,2,-2,2);
        glMatrixMode(GL_MODELVIEW);
}
void disp_tri(float a[3],float b[3],float c[3])
{
        glBegin(GL_POLYGON);
                glVertex3fv(a);
                glVertex3fv(b);
                glVertex3fv(c);
        glEnd();
}
void div_tri(float *a,float *b,float *c,int n)
```

```
{
        float v1[3],v2[3],v3[3];
        int i;

        if(n>0)
        {
                for(i=0;i<3;i++)
                {
                        v1[i] = (a[i] + b[i])/2;
                        v2[i] = (a[i] + c[i])/2;
                        v3[i] = (b[i] + c[i])/2;
                }
                div_tri(a,v1,v2,n-1);
                div_tri(v1,b,v3,n-1);
                div_tri(v2,v3,c,n-1);
        }
        else
                disp_tri(a,b,c);
}

void display()
{
        glClearColor(0,0,0,1);
        glClear(GL_COLOR_BUFFER_BIT);

        glColor3f(1,1,1);
        div_tri(v[1],v[2],v[3],c);

        glColor3f(1,0,0);
        div_tri(v[0],v[1],v[2],c);

        glColor3f(0,1,0);
        div_tri(v[0],v[2],v[3],c);

        glColor3f(0,0,1);
        div_tri(v[0],v[1],v[3],c);

        glFlush();
}

void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(600,600);
        glutInitWindowPosition(300,150);

        printf("Enter the number of subdivisions:\n");
        scanf("%d",&c);
        glutCreateWindow("3D Recursive Sierpinski Gasket");
        myinit();
        glutDisplayFunc(display);
        glutMainLoop();
}
```

13. CYLINDER AND PARALLELOPIPED

Program to create a cylinder and a parallelepiped by extruding a circle and quadrilateral respectively. Allow the user to specify the circle and the quadrilateral.

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

void myinit()
{
        glMatrixMode(GL_PROJECTION_MATRIX);
        glLoadIdentity();
        gluOrtho2D(0,200,0,200);
        glMatrixMode(GL_MODELVIEW);
}

void display()
{
        float cx=20,cy=20;
        float dy=1,cx1=50,cx2=70,cy1=20,cy2=30,dx1=0.5,dy1=0.5,x,y;
        int i,theta,r=10,n=100;

        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);

        for(i=0;i<n;i++)
        {
                glBegin(GL_POINTS);
                for(theta=0;theta<=360;theta+=1)
                {
                        x = cx + r * cos(3.142 * theta/180);
                        y = cy + r * sin(3.142 * theta/180);
                        glVertex2f(x,y);
                }
                glEnd();
                cy += dy;
        }
        glFlush();

        for(i=0;i<n;i++)
```

```
	{
		glBegin(GL_LINE_LOOP);
			glVertex2f(cx1,cy1);
			glVertex2f(cx2,cy1);
			glVertex2f(cx2,cy2);
			glVertex2f(cx1,cy2);
		glEnd();
		cx1 += dx1;
		cx2 += dx1;
		cy1 += dy1;
		cy2 += dy1;
	}
	glFlush();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(600,600);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Circle and Rectangle");
	myinit();
	glutDisplayFunc(display);
	glutMainLoop();
}
```

14. IDLE FUNCTION

PROGRAM TO DEMONSTRATE IDLE FUNCTION.

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

int theta=0;
float x,y,r=50;

void init()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(-100,100,-100,100);
	glMatrixMode(GL_MODELVIEW);
}

void idle()
{
	theta +=3;
	if(theta >= 360)
		theta=0;
```

```
        for(unsigned long int i=0;i<=50000000;i++);
        glutPostRedisplay();
}

void disp()
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0,0,1);

        x = r * cos(theta*3.142/180);
        y = r * sin(theta*3.142/180);

        glBegin(GL_POLYGON);
                glVertex2f(x,y);
                glVertex2f(-1*y,x);
                glVertex2f(-1*x,-1*y);
                glVertex2f(y,-1*x);
        glEnd();
        glutSwapBuffers();
        glFlush();
}

void mouse(int b,int s,int x,int y)
{
        if(b==GLUT_LEFT_BUTTON && s==GLUT_DOWN)
                glutIdleFunc(idle);
        if(b==GLUT_RIGHT_BUTTON && s==GLUT_DOWN)
                glutIdleFunc(NULL);
}

void main()
{
        glutInitDisplayMode(GLUT_DOUBLE| GLUT_RGB);
        glutInitWindowSize(600,600);
        glutInitWindowPosition(300,150);
        glutCreateWindow("Idle");
        init();

        glutDisplayFunc(disp);
        glutMouseFunc(mouse);
        glutIdleFunc(idle);
        glutMainLoop();
}
```

15. **RESHAPE FUNCTION**
    PROGRAM TO DEMONSTRATE RESHAPE FUNCTION.

```
#include<stdio.h>
```

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

void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(-100,100,-100,100);
	glMatrixMode(GL_MODELVIEW);
}

void display()
{
	glClearColor(0,0,0,1);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
		glBegin(GL_POLYGON);
			glVertex2f(-50,-50);
			glVertex2f(-50,50);
			glVertex2f(50,50);
			glVertex2f(50,-50);
		glEnd();

	glFlush();
}

void reshape(int w,int h)
{
	glViewport(0,0,w,h);
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();

	float t1 = (float)w/(float)h;
	float t2 = (float)h/(float)w;

	if(w>h)
		gluOrtho2D(-100*t1,100*t1,-100,100);
	else
		gluOrtho2D(-100,100,-100*t2,100*t2);
	glMatrixMode(GL_MODELVIEW);
	glutPostRedisplay();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(200,200);
	glutInitWindowPosition(300,150);
```

```
        glutCreateWindow("Reshape");
        myinit();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
}
```

16. MENU OPTION

PROGRAM TO DEMONSTRATE MENU FUNCTION.

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

int w=640,h=480;
int sizef=0;

void init()
{
        glMatrixMode(GL_PROJECTION_MATRIX);
        glLoadIdentity();
        gluOrtho2D(0,w,0,h);
        glMatrixMode(GL_MODELVIEW);
}

void disp()
{
        glClearColor(1,1,1,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0,0,1);
        glBegin(GL_POLYGON);
                glVertex2f(250-sizef,250-sizef);
                glVertex2f(350+sizef,250-sizef);
                glVertex2f(350+sizef,350+sizef);
                glVertex2f(250-sizef,350+sizef);
        glEnd();
        glFlush();
}

void d_menu(int op)
{
        if(op==1)
                sizef +=5;
        else if(op==2)
                sizef -=5;
        else if(op==3)
                sizef =0;
```

```
	else if(op==4)
		exit(0);
	glutPostRedisplay();
}

void keyboard(unsigned char key,int x,int y)
{
	if(key=='i')
		sizef +=5;
	else if(key=='d')
		sizef -=5;
	else if(key=='r')
		sizef =0;
	else if(key=='q')
		exit(0);
	glutPostRedisplay();
}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(w,h);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Menu");
	init();

	glutCreateMenu(d_menu);
		glutAddMenuEntry("Increase",1);
		glutAddMenuEntry("Deccrease",2);
		glutAddMenuEntry("Refresh",3);
		glutAddMenuEntry("Quit",4);
	glutAttachMenu(GLUT_RIGHT_BUTTON);
	glutDisplayFunc(disp);
	glutKeyboardFunc(keyboard);
	glutMainLoop();
}
```

17. MOUSE FUNCTION

PROGRAM TO DEMONSTRATE MOUSE FUNCTION.

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

int w=640,h=480;
int side=40;

void init()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(0,w,0,h);
	glMatrixMode(GL_MODELVIEW);
}

void disp()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);
	//glColor3f(0,0,1);
	glFlush();
}

void mousef(int b,int s,int x,int y)
{
	float t;
	y=h-y;
	if(b==GLUT_LEFT_BUTTON && s==GLUT_DOWN)
	{
		t=side/2.0;
		glColor3f(0,0,1);
		glBegin(GL_POLYGON);
			glVertex2f(x+t,y+t);
			glVertex2f(x+t,y-t);
			glVertex2f(x-t,y-t);
			glVertex2f(x-t,y+t);
		glEnd();
		glFlush();
	}

	if(b==GLUT_RIGHT_BUTTON && s==GLUT_DOWN)
	{	glClearColor(1,1,1,1);
		glClear(GL_COLOR_BUFFER_BIT);
		glFlush();
	}

}

void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(w,h);
	glutInitWindowPosition(300,150);
	glutCreateWindow("Menu");
	init();
	glutDisplayFunc(disp);
	glutMouseFunc(mousef);
	glutMainLoop();
}
```

18. TEXT

PROGRAM TO DISPLAY TEXT.

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

char str[11];
int cx1=50,cy1=100,cx2=20,cy2=180,d=10;

void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(0,200,0,200);
	glMatrixMode(GL_MODELVIEW);
}

void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRasterPos2i(cx1,cy1);

	int i;

	glRasterPos2i(cx1,cy1);
	for(i=0;i<strlen(str);i++)
	{

		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
	}

	glColor3f(0,0,1);
	for(i=0;i<strlen(str);i++)
	{
		glRasterPos2i(cx2,cy2-d*i);
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
	}

	glFlush();
}
```

```
void main()
{
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(300,150);

	printf("Enter the string:\n");
	gets(str);

	glutCreateWindow("Text");
	myinit();
	glutDisplayFunc(display);
	glutMainLoop();
}
```

19. MOVE TEXT

PROGRAM TO DEMONSTRATE TO MOVE TEXT.

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

char str[11];
int cx1=50,cy1=100,cx2=20,cy2=180,d=10;

void myinit()
{
	glMatrixMode(GL_PROJECTION_MATRIX);
	glLoadIdentity();
	gluOrtho2D(0,200,0,200);
	glMatrixMode(GL_MODELVIEW);
}

void display()
{
	glClearColor(1,1,1,1);
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3f(1,0,0);
	glRasterPos2i(cx1,cy1);

	int i,j;
	long int k;

	for(j=0;j<100;j++)
	{
		glRasterPos2i(cx1+5*j,cy1);
```

```
			for(i=0;i<strlen(str);i++)
				glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);

			glFlush();
			for(k=0;k<6500000;k++);
			glClear(GL_COLOR_BUFFER_BIT);
			glColor3f(1,0,0);
		}

		glFlush();
}

void main()
{
		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(300,150);

		printf("Enter the string:\n");
		gets(str);

		glutCreateWindow("Text");
		myinit();

		glutDisplayFunc(display);

		glutMainLoop();
}
```

20. DISPLAY INTEGER

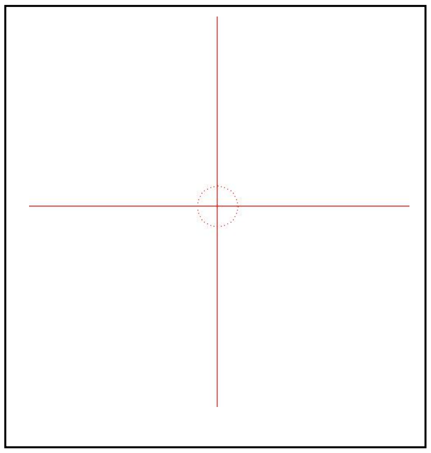
prog to display number of any digit

```
#include<stdlib.h>
#include<GL\glut.h>
#include<stdio.h>
#include<string.h>
#include<math.h>
int n;
void myinit()
{
		glMatrixMode(GL_PROJECTION);
		glLoadIdentity();
		gluOrtho2D(0,100,0,100);
		glMatrixMode(GL_MODELVIEW);
}

void display()
{
			int i;
			char str[20];
```

```
                glClearColor(1,1,1,1);
                glClear(GL_COLOR_BUFFER_BIT);

                _itoa_s(n,str,10);//converting integer to its acscii and storing it in string str
                //counting the num of digits

                glColor3f(0,1,0);//defining the color of the primitive here its num
                glRasterPos2f(10,90);
                for(i=0;i<strlen(str);i++)
                {
                        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,str[i]);
                }


                glFlush();
}
int main()
{
                glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
                glutInitWindowPosition(100,100);
                glutInitWindowSize(500,500);
                glutCreateWindow("N DIGIT NUMBER");
                printf("Enter a num");
                scanf_s("%d",&n);//accepting the number
                myinit();
                glutDisplayFunc(display);
                glutMainLoop();

}
```

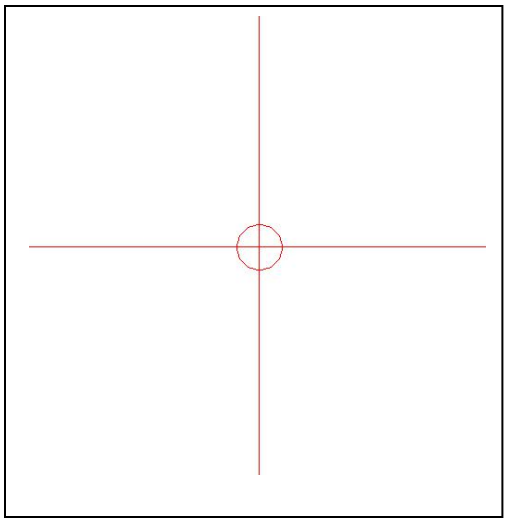
// openGL program to draw a circle using points

```
#include<stdlib.h>
#include<math.h>
#include<GL/glut.h>

void myinit()
{


        gluOrtho2D(-100, 100, -100, 100);

}

void display()
{
        glClearColor(1, 1, 1, 1);
        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double angle;
```



```
        glColor3f(1, 0, 0);
        // to display all 4 quadrants
        glBegin(GL_LINES);
                glVertex2i(0, 100);
                glVertex2i(0,-100);
                glVertex2i(100,0);
                glVertex2i(-100,0);
        glEnd();

        glBegin(GL_POINTS);
        for (i = 0; i <= 36; i++) // each points at 10 degree
        {
                angle = 3.14 / 18 * i;
                glVertex2f(10 * cos(angle), 10 * sin(angle)); // to display in center
                //glVertex2f(10 * cos(angle) + 50, 10 * sin(angle) +50); //to display in 1st quadrant
                //glVertex2f(10 * cos(angle) - 50, 10 * sin(angle) +50); //to display in 2ndth quadrant
                //glVertex2f(10 * cos(angle) -50, 10 * sin(angle) - 50); to display in 3rd quadrant
                //glVertex2f(10 * cos(angle)+50, 10 * sin(angle)-50); to display in 4th quadrant
                //glVertex2f(15 * cos(angle), 15 * sin(angle));
        }
        glEnd();
        glFlush();
}

void main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow(" circle using points");
                myinit();
        glutDisplayFunc(display);
        glutMainLoop();
    }
```

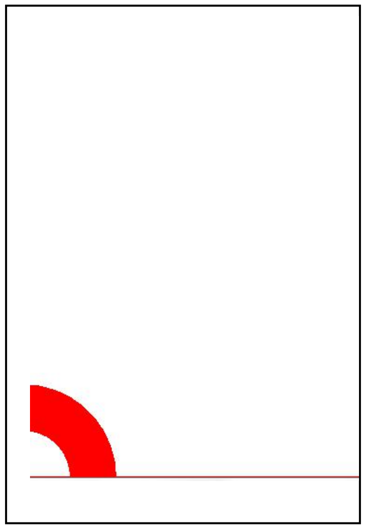
//Pgm to draw circle using lines



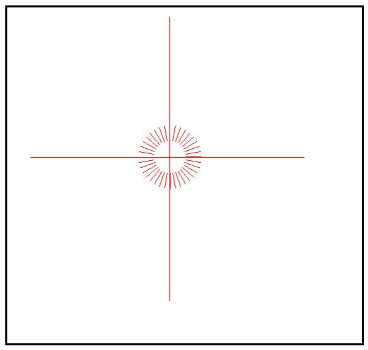
```
    void display()
    {
        glClearColor(1, 1, 1, 1);
        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double angle;
        glColor3f(1, 0, 0);
        // to display all 4 quadrants
        glBegin(GL_LINES)
                glVertex2i(0, 100);
                glVertex2i(0,-100);
                glVertex2i(100,0);
                glVertex2i(-100,0);
        glEnd();
        // to draw circle, coordinate points at 30 degree
        glBegin(GL_LINE_LOOP);
        for (i = 0; i <= 12; i++)
        {
        angle = 3.14 / 6 * i;
        glVertex2f(10 * cos(angle), 10 * sin(angle)); // to display in center
        //glVertex2f(10 * cos(angle) + 50, 10 * sin(angle) +50); //to display in 1st quadrant
        //glVertex2f(10 * cos(angle) - 50, 10 * sin(angle) +50); //to display in 2ndth quadrant
        //glVertex2f(10 * cos(angle) -50, 10 * sin(angle) - 50); to display in 3rd quadrant
        //glVertex2f(10 * cos(angle)+50, 10 * sin(angle)-50); to display in 4th quadrant
        //glVertex2f(15 * cos(angle), 15 * sin(angle));
        }
        glEnd();
        glFlush();
    }

    void main()
    {
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("circle using LINES");

        myinit();
        glutDisplayFunc(display);
        glutMainLoop();
    }
```

PGM to draw circle using quad strip/ to draw letter ‘O’

```
#include<stdlib.h>
#include<math.h>
#include<GL/glut.h>

void myinit()
{

       gluOrtho2D(0, 100, 0, 100); // does not cover all points in 360 degree
       // as x and y values change to –ve also. So change to below statement.
       //gluOrtho2D(-100, 100, -100, 100);

}

void display()
{
       glClearColor(1, 1, 1, 1);
       glClear(GL_COLOR_BUFFER_BIT);
       int i;
       double angle;
       glColor3f(1, 0, 0);
       // to display all 4 quadrants
       glBegin(GL_LINES);
              glVertex2i(0, 100);
              glVertex2i(0,-100);
              glVertex2i(100,0);
              glVertex2i(-100,0);
       glEnd();

       glBegin(GL_QUAD_STRIP);
       for (i = 0; i <= 36; i++)
       {
              angle = 3.14 / 18 * i; // 10 degree angle
              glVertex2f(10 * cos(angle), 10 * sin(angle));
              glVertex2f(20 * cos(angle), 20 * sin(angle));
       }
       glEnd();
       glFlush();
}

void main()
{
       glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
       glutInitWindowSize(500, 500);
       glutInitWindowPosition(100, 100);
       glutCreateWindow(" circle using points");
              myinit();
       glutDisplayFunc(display);
       glutMainLoop();
   }
```



```
#include<stdlib.h>
#include<math.h>
#include<stdio.h>
#include<GL/glut.h>
void myinit()
{

        //gluOrtho2D(0, 100, 0, 100);
        gluOrtho2D(-100, 100, -100, 100);

}

void display()
{
        glClearColor(1, 1, 1, 1);
        glClear(GL_COLOR_BUFFER_BIT);
        int i;
        double angle;
        glColor3f(1, 0, 0);
```



```
        // to display all 4 quadrants
        glBegin(GL_LINES);
                glVertex2i(0, 100);
                glVertex2i(0,-100);
                glVertex2i(100,0);
                glVertex2i(-100,0);
        glEnd();

        glBegin(GL_LINES);
        for (i = 0; i <= 36; i++)
        {
                angle = 3.14 / 18 * i;
                glVertex2f(10 * cos(angle), 10 * sin(angle));
                glVertex2f(20 * cos(angle), 20 * sin(angle)); // to display in center

        }
        glEnd();
        glFlush();
}

int main()
{
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);
        glutCreateWindow(" circle using Lines");
                myinit();
                        glutDisplayFunc(display);
        glutMainLoop();
        return 0;
    }
```

```
// interactive dda
#include <math.h>
#include<stdlib.h>
#include <glut.h>

#include <iostream>
using namespace std;

struct Point {
        GLint x;
        GLint y;
};

Point p1, p2;

void draw_dda(Point p1, Point p2) {
        GLfloat dx = p2.x - p1.x;
        GLfloat dy = p2.y - p1.y;
        GLfloat x1 = p1.x;
        GLfloat y1 = p1.y;

        GLfloat step = 0;

        if (abs(dx) > abs(dy)) {
               step = abs(dx);
        }
        else {
               step = abs(dy);
        }

        GLfloat xInc = dx / step;
        GLfloat yInc = dy / step;

        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_POINTS);
        for (float i = 1; i <= step; i++) {
               glVertex2i(x1, y1);
               x1 += xInc;
               y1 += yInc;
        }
        glEnd();
        glFlush();
}

void myMouseFunc(int button, int state, int x, int y)
{
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
               p1.x = x;
               p1.y = 480 - y;
        }
        else if (button == GLUT_LEFT_BUTTON && state == GLUT_UP) {
               p2.x = x;
               p2.y = 480 - y;
               draw_dda(p1, p2);
        }
}

void init() {
        glClearColor(1.0, 1.0, 1.0, 0.0);
        glColor3f(0.0f, 0.0f, 0.0f);
        glPointSize(1.0f);
        gluOrtho2D(0.0f, 640.0f, 0.0f, 480.0f);
        //gluOrtho2D(0, 500, 0, 500);
}

void display(void) {}

int main(int argc, char **argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowPosition(200, 200);
        //glutInitWindowSize(640, 480);
        glutInitWindowSize(500, 500);
        glutCreateWindow("Mouse Func");
        glutDisplayFunc(display);
        glutMouseFunc(myMouseFunc);
        init();
        glutMainLoop();

        return 0;
    }



    // hierarchical menu

#include<stdlib.h>
#include<math.h>
#include<stdio.h>
#include<glut.h>
GLsizei wh = 500, ww = 500;
GLfloat size = 3.0;

void display()
{
        glFlush();
}

void myInit()
{
        glViewport(0, 0, ww, wh);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, (GLdouble)ww, 0.0, (GLdouble)wh);
        glMatrixMode(GL_MODELVIEW);
        glClearColor(0.0, 0.0, 0.0, 1.0);
        glColor3f(1.0, 0.0, 0.0);
}

void myReshape(GLsizei w, GLsizei h)
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glViewport(0, 0, w, h);
        ww = w;
        wh = h;
}

void drawSquare(int x, int y)
{
        y = wh - y;
        glBegin(GL_POLYGON);
        glVertex2f(x + size, y + size);
        glVertex2f(x - size, y + size);
        glVertex2f(x - size, y - size);
        glVertex2f(x + size, y - size);
        glEnd();
        glFlush();
}
```

```
void size_menu(int id)
{
	switch (id)
	{
	case 2: size = size * 2;
		break;
	case 3:if (size > 1) size = size / 2;
		break;
	}
	glutPostRedisplay();
}

void top_menu(int id)
{
	switch (id)
	{
	case 1:exit(0); break;
	}
	glutPostRedisplay();
}

void myMouse(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
		drawSquare(x, y);
	if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
		exit(0);
    }
```