```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

float rtri =-1.0f;

void InitGL(int Width,int Height)
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
  glClearDepth(1.0);
  glDepthFunc(GL_LESS);
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_SMOOTH);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}

void ReSizeGLScene(int Width, int Height)
{
  if (Height==0)
    Height=1;
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}


void drawRect()
{
        glBegin(GL_POLYGON);
                glColor3f(1.0f,0.0f,0.0f);
                glVertex2f(-1.1f, 1.0f);
                glVertex2f(-1.0f, 1.0f);
                glVertex2f(-1.1f,-10.0f);

                glVertex2f(-1.0f, 1.0f);
                glVertex2f(-1.0f, -10.0f);
                glVertex2f(-1.1f,-10.0f);
        glEnd();
        glFlush();
}

void DrawGLScene()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glTranslatef(0.0f,rtri,-6.0f);
        glBegin(GL_POLYGON);
                glColor3f(1.0f,0.3f,0.0f);
```

```
                glVertex2f(-1.0f, 1.0f);
                glVertex2f(0.4f, 1.0f);
                glVertex2f(-1.0f,0.0f);

                glVertex2f(0.4f, 1.0f);
                glVertex2f(0.4f, 0.0f);
                glVertex2f(-1.0f,0.0f);
        glEnd();
        rtri+=0.005f;
        if(rtri>1)
                rtri-=0.005f;
        drawRect();
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutInitWindowSize(640, 480);
  glutInitWindowPosition(0, 0);
  glutCreateWindow("flagHoistingBETAver1.0");
  glutDisplayFunc(&DrawGLScene);
  glutIdleFunc(&DrawGLScene);
  glutReshapeFunc(&ReSizeGLScene);
  InitGL(640, 480);
  glutMainLoop();
  return 0;
}
```

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

int xx1,xx2,xx3,yy1,yy2,yy3;

void myInit(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(0.0,0.0,0.0,1.0);
        glMatrixMode(GL_PROJECTION);
        gluOrtho2D(0,500,0,500);
}

void draw_pixel(int x, int y)
{
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
}

void draw_line(int x1, int x2, int y1, int y2)
{
        int dx, dy, i=0, p;
        int incx=1, incy=1, inc1, inc2;
        int x,y;
        dx = abs(x2-x1);
        dy = abs(y2-y1);
        if (x2 < x1)
                incx = -1;
        if (y2 < y1)
                incy = -1;
        x = x1;
        y = y1;
        if (dx > dy)
        {
                draw_pixel(x, y);
                p = (2*dy)-dx;
                inc1 = 2*(dy-dx);
                inc2 = 2*dy;
                while(i<dx)
                {
                        if (p >= 0)
                        {
                                y += incy;
                                p += inc1;
                        }
                        else
                                p += inc2;
                        x += incx;
                        draw_pixel(x, y);
```

```
                        i++;
                }
        }
        else
        {
                draw_pixel(x, y);
                p = (2*dx)-dy;
                inc1 = 2*(dx-dy);
                inc2 = 2*dx;
                while(i<dy)
                {
                        if (p >= 0)
                        {
                                x += incx;
                                p += inc1;
                        }
                        else
                                p += inc2;
                        y += incy;
                        draw_pixel(x, y);
                        i++;
                }
        }
}


void draw_circle(int cx,int cy,int r)
{
        int x=0,y=r,p=3-(2*r);
        int inc1, inc2;
        glBegin(GL_POINTS);
        for(int i=x;i<y;i++)
        {
                inc1=4*x+6;
                inc2=4*(x-y)+10;
                if(p<0)
                {
                        p+=inc1;
                        x+=1;
                }
                else
                {
                        y-=1;
                        x++;
                        p+=inc2;
                }
                glVertex2i(cx+x,cy+y);
                glVertex2i(cx+y,cy+x);
                glVertex2i(cx+y,cy-x);
                glVertex2i(cx+x,cy-y);
                glVertex2i(cx-x,cy-y);
                glVertex2i(cx-y,cy-x);
```

```
                glVertex2i(cx-y,cy+x);
                glVertex2i(cx-x,cy+y);

        }
        glEnd();
}

void myDisplay()
{
        draw_line(xx1,xx2,yy1,yy2);
        draw_line(xx2,xx3,yy2,yy3);
        draw_line(xx3,xx1,yy3,yy1);

        int cy=(yy1+yy2+yy3)/3;
        int cx=(xx2+xx3+xx1)/3;
        int dx=(xx2+xx3)/2;
        int dy=(yy2+yy3)/2;
        int r=sqrt((cx-dx)*(cx-dx) + (cy-dy)*(cy-dy) );
        draw_circle(cx,cy,r);

        r=sqrt((cx-xx2)*(cx-xx2) + (cy-yy2)*(cy-yy2));
        draw_circle(cx,cy,r);
        glFlush();
}

int main(int argc, char** argv)
{
    in:
        cout<<"<<<...Enter Coordinates of a Equilateral triangle...>>>"<<endl;
        cout<<"Enter 1st Vertex Points :";
        cin>>xx1>>yy1;
        cout<<"Enter 2nd Vertex Points : ";
        cin>>xx2>>yy2;
        cout<<"Enter 3rd Vertex Points : ";
        cin>>xx3>>yy3;
        int d1=sqrt((xx1-xx2)*(xx1-xx2) + (yy1-yy2)*(yy1-yy2));
        int d2=sqrt((xx2-xx3)*(xx2-xx3) + (yy2-yy3)*(yy2-yy3));
        int d3=sqrt((xx3-xx1)*(xx3-xx1) + (yy3-yy1)*(yy3-yy1));
        if(abs(d1-d2)<=2 && abs(d2-d3)<=2 && abs(d3-d1)<=2)
    {
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(0,0);
        glutCreateWindow("Ujjal Bresenham's Circle Drawing");
        myInit();
        glutDisplayFunc(myDisplay);
        glutMainLoop();
    }
    else{
        cout<<"!!!...Enter Coordinates of a Equilateral triangle...!!!"<<endl;
        goto in;
```

```
                                        2(1)
    }
        return 0;
}
```

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;
int r=0,g=0,b=0;
vector<int> x_coordinates,y_coordinates;
int f=0;

void init2D()
{
        glClearColor(0.3,0.7,0.3,1.0);
        glPointSize(3);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,640,0,480);
}

void palette(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,0.0); //yellow
        glRecti(10,10,30,30);
        glColor3f(1.0,0.0,0.0); //red
        glRecti(30,10,50,30);
        glColor3f(0.0,1.0,1.0); //cyan
        glRecti(10,30,30,50);
        glColor3f(0.0,0.0,1.0); //blue
        glRecti(30,30,50,50);
        glColor3f(1.0,1.0,1.0); //white
        glRecti(10,50,30,70);
        glColor3f(0.0,1.0,0.0); //green
        glRecti(30,50,50,70);
        glColor3f(1.0,0.0,1.0); //magenta
        glRecti(10,70,50,90);
        glColor3f(0.0,0.0,0.0); //black
        glRecti(30,70,50,90);

        glFlush();
}

void drawPixel(int x,int y)
{
      glColor3f(r,g,b);
        glBegin(GL_POINTS);
                glVertex2i(x,480-y);
        glEnd();
        glFlush();
}

/*void getPixel(int x,int y,GLubyte *color)
{
        glReadPixels(x,y,1,1,GL_RGB,GL_UNSIGNED_BYTE,color);
}*/
```

```
void dda(double x1,double y1,double x2,double y2)
{
        double dx=(x2-x1);
        double dy=(y2-y1);
        double len;
        float xInc,yInc,x=x1,y=y1;
        len=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
        xInc=dx/(float)len;
        yInc=dy/(float)len;
        drawPixel(round(x),round(y));
        int k;
        for(k=0;k<len;k++)
        {
                x+=xInc;
                y+=yInc;
                drawPixel(round(x),round(y));
        }
}
void rotate(int x)
{cout<<x;
}
void translate(int x)
{
cout<<-x;
}
void menu()
{
        int id[20];
        id[2]=glutCreateMenu(rotate);
        glutAddMenuEntry("30",30);
        glutAddMenuEntry("45",45);
        glutAddMenuEntry("56",56);
        id[1]=glutCreateMenu(translate);
        glutAddMenuEntry("30",30);
        glutAddMenuEntry("45",45);
        glutAddMenuEntry("56",56);
        id[0]=glutCreateMenu(NULL);
        glutAddSubMenu("translate",id[1]);
        glutAddSubMenu("scale",id[2]);
        glutAttachMenu(GLUT_MIDDLE_BUTTON);


}
void color(int x,int y)
{
     if(x>=10&& x<=30)
     {
          if(y>=10&& y<=30)
          {
             r=1; g=1;b=0;
          }
```

```
        if(y>=30&&y<=50)
         {
                 r=0;g=1;b=1;
         }
         if(y>=50&&y<=70)
         {
                 r=1;g=1;b=1;
         }
         if(y>=70&&y<=90)
         {
                 r=01;g=0;b=1;
         }
    }
    if(x>30&&x<=50)
    {

        if(y>=10&& y<=30)
        {
           r=1; g=0;b=0;
        }
        if(y>=30&&y<=50)
         {
                 r=0;g=0;b=1;
         }
         if(y>=50&&y<=70)
         {
                 r=0;g=1;b=0;
         }
         if(y>=70&&y<=90)
         {
                 r=0;g=0;b=0;
         }
    }
}
void mouse(int button,int state,int x,int y)
{
        color(x,480-y);

        if(state==GLUT_DOWN)
        {
                if(button==GLUT_LEFT_BUTTON)
                {
                        if(f==1)
                        {
                                x_coordinates.clear();
                                y_coordinates.clear();
                                f=0;
                        }
                        drawPixel(x,y);
                        x_coordinates.push_back(x);
                        y_coordinates.push_back(y);
                }
```

```
                                3(1)
                else if(button==GLUT_RIGHT_BUTTON)
                {
                        GLubyte rgb[3];
                        //getPixel(x,480-y,rgb);
                        if(rgb[0]==0.0 && rgb[1]==0.0 && rgb[2]==0.0)
                        {
                                for(int i=0;i<x_coordinates.size();i++)
                                {
                                        if(i==x_coordinates.size()-1)

dda(x_coordinates[i],y_coordinates[i],x_coordinates[0],y_coordinates[0]);
                                        else

dda(x_coordinates[i],y_coordinates[i],x_coordinates[i+1],y_coordinates[i+1]);
                                }
                                f=1;
                        }
                        //else
                                //glColor3f(rgb[0],rgb[1],rgb[2]);
                }

        }
}

void keyboard(unsigned char key,int x,int y)
{
        if(key==27)
                exit(0);
}

int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE);
        glutInitWindowSize(640,480);
        glutInitWindowPosition(0,0);
        glutCreateWindow("assign1");
        init2D();
        menu();
        glutDisplayFunc(palette);
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

int ww=500,wh=500;
float x11,y11,x22,y22;
float boundaryCol[3]={0.0,0.0,0.0};
float fillCol[3]={1.0,0.0,0.0};
int
size=8,x_arr[8]={204,275,134,207,277,345,204,276},y_arr[8]={348,348,282,277,277,
277,207,207};

void setPixel(int x,int y,float fcol[3])
{
        glPointSize(2);
        glBegin(GL_POINTS);
                glColor3fv(fcol);
                glVertex2i(x,y);
        glEnd();
        glFlush();
}

void getPixel(int x,int y,float *color)
{
    glReadPixels(x,y,1,1,GL_RGB,GL_FLOAT,color);
}

void dda(double x1,double y1,double x2,double y2)
{
        double dx=(x2-x1);
        double dy=(y2-y1);
        double len;
        float xInc,yInc,x=x1,y=y1;
        len=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));
        xInc=dx/(float)len;
        yInc=dy/(float)len;
        setPixel(round(x),round(y),boundaryCol);
        int k;
        for(k=0;k<len;k++)
        {
                x+=xInc;
                y+=yInc;
                setPixel(round(x),round(y),boundaryCol);
        }
}

void rotate(int a,int b,int c,int d)
{
        x11=a*cos(0.785398)-b*sin(0.785398);
        y11=a*sin(0.785398)+b*cos(0.785398);
        x22=c*cos(0.785398)-d*sin(0.785398);
        y22=c*sin(0.785398)+d*cos(0.785398);
```

```
}

void translate(int a,int b,int c,int d)
{
        x11=a+100;
        y11=b-75;
        x22=c+100;
        y22=d-75;
}

void display()
{
        glClearColor(1.0,1.0,1.0,0);
        glLoadIdentity();
        gluOrtho2D(0.0,ww,0.0,wh);
        glClear(GL_COLOR_BUFFER_BIT);

        rotate(250,50,450,50);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(450,50,450,250);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(450,250,250,250);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(250,250,250,50);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);

        rotate(300,50,300,250);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(350,50,350,250);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(400,50,400,250);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);

        rotate(250,100,450,100);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(250,150,450,150);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
        rotate(250,200,450,200);
        translate(x11,y11,x22,y22);
        dda(x11,y11,x22,y22);
}

void floodfill(int x,int y,float oldcolor[3],float newcolor[3])
```

```
{
        float color[3];
        getPixel(x,y,color);
        //cout<<x<<" "<<y<<endl;
        //cout<<color[0]<<" "<<color[1]<<" "<<color[2]<<endl;
        if(color[0] && color[1] && color[2])
        {
                setPixel(x,y,newcolor);
                floodfill(x+1,y,oldcolor,newcolor); //4 point
                floodfill(x-2,y,oldcolor,newcolor); //w
                floodfill(x,y+1,oldcolor,newcolor);
                floodfill(x,y-2,oldcolor,newcolor);
                /*floodfill(x+2,y+2,oldcolor,newcolor); //8 point
                floodfill(x-2,y+2,oldcolor,newcolor);
                floodfill(x+2,y-2,oldcolor,newcolor);
                floodfill(x-2,y-2,oldcolor,newcolor);*/
                //setPixel(x,y,newcolor);
        }
}

void mouse(int btn,int state,int x,int y)
{
        if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        {
                int xi=x;
                int yi=(wh-y);
                //cout<<xi<<" "<<yi<<endl;
                for(int i=0;i<size;i++)
                        floodfill(x_arr[i],y_arr[i],boundaryCol,fillCol);
        }
}

int main(int argc,char** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE);
        glutInitWindowSize(ww,wh);
        glutCreateWindow("assign4");
        glutDisplayFunc(display);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```

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

const int w=500,h=500;
int flag=1,wi[4][2],wo[4][2],n=0,m=0;
int ymax, ymin, xmax, xmin;
void *font[]={GLUT_BITMAP_8_BY_13,GLUT_BITMAP_HELVETICA_18 };

typedef struct point{
        int xc,yc;
}point;
point e[20],out[20] ;

void set(int x, int y)
{
        glColor3f(1,0,0);
        glPointSize(4);
        glBegin(GL_POINTS);
        glVertex2f(x,y);
        glEnd();
        glFlush();
}
void abc()
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,w,0,h);
}

void menu()
{
        glColor3f(1,1,1);
        glRasterPos2f(100,480);

        char m[]="Pls read carefully Instruction Before Starting:::";
        for(int i=0;i<50;i++)
        glutBitmapCharacter(font[1],m[i]);
        glEnd();
        glRasterPos2f(10,400);
        char m1[]="1...First click by left button in upper window twice";
                char m2[]="2...right click for window creating";
                for(int i=0;i<55;i++)
                glutBitmapCharacter(font[0],m1[i]);
                glEnd();
                glRasterPos2f(10,350);
                for(int i=0;i<36;i++)
                                glutBitmapCharacter(font[0],m2[i]);
                glEnd();
                char m3[]="3.....Now draw polygon with left click...";
                glRasterPos2f(10,300);
                for(int i=0;i<45;i++)
```

```
                        glutBitmapCharacter(font[0],m3[i]);
                glEnd();
                char m4[]="START";
                                glRasterPos2f(253,255);
                                for(int i=0;i<6;i++)
                                glutBitmapCharacter(font[0],m4[i]);
                                glEnd();
}
void desplay()
{
        glColor3f(1,1,0);
                glPointSize(4);
        glClearColor(0.25,.13,.16,0);
        glClear(GL_COLOR_BUFFER_BIT);
        menu();
        glColor3f(1,1,1);
                glBegin(GL_LINE_LOOP);
                glVertex2d(250,250);
                glVertex2d(250,270);
                glVertex2d(300,270);
                glVertex2d(300,250);
                glEnd();
                glFlush();




}

void clip()
{
        float s;
        int c=0;
        for(int i=0;i<m;i++)
        {
                if(e[i].xc<xmin&&e[i+1].xc>xmin)
                {
                        s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);
                        out[c].yc=e[i].yc+(int)(s*(xmin-e[i].xc));
                        out[c].xc=xmin;
                        c++;
                        out[c].xc=e[i+1].xc;
                        out[c].yc=e[i+1].yc;
                        c++;
                }
                else if(e[i].xc>xmin&&e[i+1].xc>xmin)
                {
                        out[c].xc=e[i+1].xc;
                        out[c].yc=e[i+1].yc;
                        c++;

                }
```

```
                else if(e[i].xc>xmin&&e[i+1].xc<xmin)
                {
                        s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);
                        out[c].yc=e[i].yc+(int)(s*(xmin-e[i].xc));
                        out[c].xc=xmin;
                        c++;
                }
        }

        out[c]=out[0];
        m=c;
        c=0;

        for(int i=0;i<m;i++)
                {
                        if(out[i].yc<ymin&&out[i+1].yc>ymin)
                        {

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);
                                e[c].xc=out[i].xc+(int)((ymin-out[i].yc)/s);
                                e[c].yc=ymin;
                                c++;
                                e[c].xc=out[i+1].xc;
                                e[c].yc=out[i+1].yc;
                                c++;
                        }
                        else if(out[i].yc>ymin&&out[i+1].yc>ymin)
                        {
                                e[c].xc=out[i+1].xc;
                                e[c].yc=out[i+1].yc;
                                c++;

                        }
                        else if(out[i].yc>ymin&&out[i+1].yc<ymin)
                        {

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);
                                e[c].xc=out[i].xc+(int)((ymin-out[i].yc)/s);
                                e[c].yc=ymin;
                                c++;
                        }
                }

        e[c]=e[0];
        m=c;
        c=0;

        for(int i=0;i<m;i++)
                {
                        if(e[i].xc>xmax&&e[i+1].xc<xmax)
                        {
```

```
s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);
                                out[c].yc=e[i].yc+(int)(s*(xmax-e[i].xc));
                                out[c].xc=xmax;
                                c++;
                                out[c].xc=e[i+1].xc;
                                out[c].yc=e[i+1].yc;
                                c++;
                        }
                        else if(e[i].xc<xmax&&e[i+1].xc<xmax)
                        {
                                out[c].xc=e[i+1].xc;
                                out[c].yc=e[i+1].yc;
                                c++;

                        }
                        else if(e[i].xc<xmax&&e[i+1].xc>xmax)
                        {

s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);
                                out[c].yc=e[i].yc+(int)(s*(xmax-e[i].xc));
                                out[c].xc=xmax;
                                c++;
                        }
                }

        out[c]=out[0];
        m=c;
        c=0;
        for(int i=0;i<m;i++)
                        {
                                if(out[i].yc>ymax&&out[i+1].yc<ymax)
                                {

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymax-out[i].yc)/s);
                                        e[c].yc=ymax;
                                        c++;
                                        e[c].xc=out[i+1].xc;
                                        e[c].yc=out[i+1].yc;
                                        c++;
                                }
                                else if(out[i].yc<ymax&&out[i+1].yc<ymax)
                                {
                                        e[c].xc=out[i+1].xc;
                                        e[c].yc=out[i+1].yc;
                                        c++;

                                }
                                else if(out[i].yc<ymax&&out[i+1].yc>ymax)
                                {
```

```
s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymax-out[i].yc)/s);
                                        e[c].yc=ymax;
                                        c++;
                                }
                        }

                e[c]=e[0];

                glColor3f(0,1,0);

        glBegin(GL_LINE_LOOP);
                for(int i=0; i<c;i++)
                        glVertex2f(e[i].xc,e[i].yc-250);
                        glEnd();
                        glFlush();



}
void draw()
{
        wi[1][0]=wi[2][0];
        wi[1][1]=wi[0][1];
        wi[3][0]=wi[0][0];
        wi[3][1]=wi[2][1];


        if(wi[0][1]>wi[3][1])
        {
                ymax=wi[0][1];
                ymin=wi[3][1];
        }
        else
        {
                ymax=wi[3][1];
                ymin=wi[0][1];

        }
        if(wi[0][0]>wi[1][0])
        {
                xmax=wi[0][0];
                xmin=wi[1][0];
        }
        else
        {
                xmax=wi[1][0];
                xmin=wi[0][0];

        }
        glRasterPos2f(180,480);
```

```
        glColor3f(0.25,0.36,0.58);
        char m[]="Befor Clipping Window";
        for(int i=0;i<26;i++)
                glutBitmapCharacter(font[0],m[i]);


        glBegin(GL_LINE_LOOP);
        for(int i=0; i<4;i++)
                glVertex2f(wi[i][0],wi[i][1]);
                glEnd();
                glFlush();


}

void drawoutput()
{
        wo[0][0]=wi[0][0];
        wo[1][0]=wi[1][0];
        wo[2][0]=wi[2][0];
        wo[3][0]=wi[3][0];
        wo[0][1]=wi[0][1]-250;
                wo[1][1]=wi[1][1]-250;
                wo[2][1]=wi[2][1]-250;
                wo[3][1]=wi[3][1]-250;

                glRasterPos2f(180,480-250);

                        char m[]="After Clipping Window";
                        for(int i=0;i<25;i++)
                                glutBitmapCharacter(font[0],m[i]);

                glPointSize(4);
                glColor3f(0,0,1);
                glBegin(GL_LINE_LOOP);
                        for(int i=0; i<4;i++)
                                glVertex2f(wo[i][0],wo[i][1]);
                                glEnd();
                                glFlush();
}

void mine(int b,int s, int x, int y)
{
        if(s==GLUT_DOWN)
        {
                if(b==GLUT_LEFT_BUTTON)
                {
                        if(flag==1)
                        {
                                wi[n][0]=x;
                                wi[n][1]=500-y;
                                n+=2;
```

```
                                        5(1)
                                set(x,500-y);
                        }
                        else
                        {
                                e[m].xc=x;
                                e[m].yc=500-y;
                                set(x, 500-y);
                                glColor3f(1,1,1);
                                if(m>0)
                                {
                                        glBegin(GL_LINES);
                                                glVertex2f(e[m-1].xc,e[m-1].yc);
                                                glVertex2f(e[m].xc, e[m].yc);
                                                glEnd();
                                                glFlush();
                                }
                                m++;
                        }
                }
                else if(b==GLUT_RIGHT_BUTTON)
                {
                        if(flag==1)
                        {
                                draw();
                                flag=0;
                        }
                        else
                        {
                                glColor3f(1,1,1);
                                e[m].xc=e[0].xc;
                                e[m].yc=e[0].yc;
                                glBegin(GL_LINES);
                                glVertex2f(e[m-1].xc, e[m-1].yc);
                                glVertex2f(e[m].xc,e[m].yc);
                                glEnd();
                                glFlush();
                                drawoutput();
                                clip();

                        }
                }
        }
}

void mine1(int b, int s, int x, int y)
{
        if(s==GLUT_DOWN)
                {
                        if(b==GLUT_LEFT_BUTTON &&(250<x &&x<300))
                        {
                                glClearColor(0.25,.13,.16,0);
                                glClear(GL_COLOR_BUFFER_BIT);
```

```
                                        5(1)
                                for(int i=0;i<500;i++)
                                        {
                                                set(i,250);
                                        }
                                glutMouseFunc(mine);
                        }
                }

}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(w,h);
        glutCreateWindow("rohit chahar: poly clip");\
        abc();
        glutDisplayFunc(desplay);
        glutMouseFunc(mine1);
        glutMainLoop();
        return 0;



}
```

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

#define w 500
#define h 500

void init()
{
        glClearColor(0.0,0.0,0.0,1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-w/2,w/2,-h/2,h/2);
}

void setpixel(GLint x, GLint y)
{
        glColor3f(1.0,1.0,1.0);
        glPointSize(2.0);
        glBegin (GL_POINTS);
                glVertex2f(x,y);
        glEnd();
        glFlush();
}

float mati[30][2],mato[30][2];
int k=0,c;

class trans
{
        int transl[2][2];
public:
        void mul(float s[2][2])
        {
                int i,j,k;
                float sum;
                for(i=0;i<c;i++)
                {
                        for(j=0;j<2;j++)
                        {
                                sum=0;
                                for(k=0;k<2;k++)
                                sum=sum+mati[i][k]*s[k][j];
                                mato[i][j]=sum;
                        }
                }
        }
        void plot(float m[][2])
        {
                glColor3f(0.0,1.0,0.0);
                glBegin(GL_LINE_LOOP);
                        for(int i=0;i<c;i++)
```

```
                                6(1)
                            glVertex2f(m[i][0],m[i][1]);
                glEnd();
                glFlush();
        }
        void translation(float tx,float ty)
        {
                int i;
                for(i=0;i<c;i++)
                {
                        mato[i][0]=mati[i][0]+ tx;
                        mato[i][1]=mati[i][1]+ ty;
                }
        }
        void scaling()
        {
                float scale[2][2];
                int i;
                float sx,sy;
                cout<<"Scaling->\nEnter Sx factor::";
                cin>>sx;
                cout<<"Enter Sy factor::";
                cin>>sy;
                scale[0][0]=sx;
                scale[0][1]=0;
                scale[1][0]=0;
                scale[1][1]=sy;
                mul(scale);
                plot(mato);
        }
        void rotation()
        {
                int rot;
                float angle,rota[2][2];
                cout<<"Rotation->\nEnter angle::";
                cin>>angle;
                angle=(3.14*angle)/180;
                cout<<"1.For Clockwise rotation\n2.For Anti-Clockwise
rotation\nEnter your choice::";
                cin>>rot;
                switch(rot)
                {
                        case 1 :rota[0][0]=cos(angle);
                                        rota[0][1]=-sin(angle);
                                        rota[1][0]=sin(angle);
                                        rota[1][1]=cos(angle);
                                        break;
                        case 2 :rota[0][0]=cos(angle);
                                        rota[0][1]=sin(angle);
                                        rota[1][0]=-sin(angle);
                                        rota[1][1]=cos(angle);
                                        break;
                        default:cout<<"Invalid Input!!!\n";
```

```
                                        6(1)
                                          system("pause");
                                          exit(0);
                }
                mul(rota);
        }
        void rot_arbit()
        {
                float xm,ym;
                int i,j;
                cout<<"Rotation about arbitrary point->\nEnter x corrdinate::";
                cin>>xm;
                cout<<"Enter y corrdinate::";
                cin>>ym;
                translation(-xm,-ym);
                for(i=0;i<c;i++)
                {
                        for(j=0;j<2;j++)
                        {
                                mati[i][j]=mato[i][j];
                        }
                }
                rotation();
                for(i=0;i<c;i++)
                {
                        for(j=0;j<2;j++)
                        {
                                mati[i][j]=mato[i][j];
                        }
                }
                translation(xm,ym);
                plot(mato);
        }
};

void menu()
{
        trans t;
        float tx,ty;
        int choice;
        do
        {
                cout<<"\n1. Translation\n2. Scaling\n3. Rotation\n4. Rotation
about arbitrary point\n5. Exit\nEnter your choice::";
                cin>>choice;
                switch(choice)
                {
                        case 1 :cout<<"Translation->\nEnter tx factor::";
                                        cin>>tx;
                                        cout<<"Enter ty factor::";
                                        cin>>ty;
                                        t.translation(tx,ty);
                                        t.plot(mato);
```

```
                                6(1)
                                  break;
                        case 2 :t.scaling();
                                  break;
                        case 3 :t.rotation();
                                  t.plot(mato);
                                  break;
                        case 4 :t.rot_arbit();
                                  break;
                        case 5 :exit(0);
                        default:cout<<"Invalid Input!!!\n";
                }
                cout<<"Want to continue(1/0)::";
                cin>>choice;
        }while(choice==1);
}

void keyboard(unsigned char key,int x,int y)
{
        if(key==27)
                exit (0);
}

void mouse(int button,int state,int x,int y)
{
        int x1,y1,p,q;
        if(state== GLUT_DOWN)
        {
                if(button==GLUT_LEFT_BUTTON)
                {
                        mati[k][0]=(float)(x-250);
                        mati[k][1]=(float)(250-y);
                        glColor3f(1.0,0.0,0.0);
                        glBegin(GL_POINTS);
                        glVertex2f(mati[k][0],mati[k][1]);
                        glEnd();
                        k++;
                        glFlush();
                }
                else if(button==GLUT_RIGHT_BUTTON)
                {
                        glBegin(GL_LINE_LOOP);
                        for(int i=0;i<k;i++)
                        glVertex2f(mati[i][0],mati[i][1]);
                        glEnd();
                        c=k;
                        k=0;
                        //menu();
                        glFlush();
                        menu();
                }

                //else if(button==GLUT_MIDDLE_BUTTON)
```

```
                                        6(1)
                        //menu();
        }
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(w,h);
        glutCreateWindow("assign6");

        glutDisplayFunc(init);
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

GLfloat angle= 0.0;

void spin (void)
{
        angle+= 1.0;
        glutPostRedisplay();
}

void display(void)
{
        glClearColor(0.3,0.7,0.3,0.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);
        glRotatef(angle, 0, 1, 0);
        glutWireCube(2.0);//glutSolidCube(double size);
        glutSwapBuffers();
}

void keyboard(unsigned char key,int x,int y)
{
        if(key==27)
                exit(0);
}

void reshape(int width,int height)
{
        glViewport(0,0,(GLsizei) width,(GLsizei) height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60,(GLfloat) width/(GLfloat) height,1.0,100.0);
        glMatrixMode(GL_MODELVIEW);
}

int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(200,100);
        glutCreateWindow("assign7");
        glutDisplayFunc(display);
        glutKeyboardFunc(keyboard);
        glutReshapeFunc(reshape);
        glutIdleFunc(spin);
        glutMainLoop();
        return 0;
}
```

----------Flag----------

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

float rtri =-1.0f;

void InitGL(int Width,int Height)
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
  glClearDepth(1.0);
  glDepthFunc(GL_LESS);
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_SMOOTH);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}

void ReSizeGLScene(int Width, int Height)
{
  if (Height==0)
    Height=1;
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}


void drawRect()
{
        glBegin(GL_POLYGON);
                glColor3f(1.0f,0.0f,0.0f);
                glVertex2f(-1.1f, 1.0f);
                glVertex2f(-1.0f, 1.0f);
                glVertex2f(-1.1f,-10.0f);

                glVertex2f(-1.0f, 1.0f);
                glVertex2f(-1.0f, -10.0f);
                glVertex2f(-1.1f,-10.0f);
        glEnd();
        glFlush();
}

void DrawGLScene()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glTranslatef(0.0f,rtri,-6.0f);
        glBegin(GL_POLYGON);
```

```
                glColor3f(1.0f,0.3f,0.0f);
                glVertex2f(-1.0f, 1.0f);
                glVertex2f(0.4f, 1.0f);
                glVertex2f(-1.0f,0.0f);

                glVertex2f(0.4f, 1.0f);
                glVertex2f(0.4f, 0.0f);
                glVertex2f(-1.0f,0.0f);
        glEnd();
        rtri+=0.005f;
        if(rtri>1)
                rtri-=0.005f;
        drawRect();
        glutSwapBuffers();
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutInitWindowSize(640, 480);
  glutInitWindowPosition(0, 0);
  glutCreateWindow("flagHoistingBETAver1.0");
  glutDisplayFunc(&DrawGLScene);
  glutIdleFunc(&DrawGLScene);
  glutReshapeFunc(&ReSizeGLScene);
  InitGL(640, 480);
  glutMainLoop();
  return 0;
}
 -------------------------Pendulum--------------------------------
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

float angle=135;
float inc=1.0;

void initialize()
{
  glClearColor(1.0,1.0,1.0,0.0);
  glClear(GL_COLOR_BUFFER_BIT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-250,250,-250,250);
}

void midPoint(int x,int y,int r)
{
  int x1=0,y1=r;
  float dp=1-r;
  while(x1<=y1)
```

```
  {
    if(dp<=0)
    {
      x1++;
      dp+=(2*x1)+1;
    }
    else
    {
      x1++;
      y1--;
      dp+=2*(x1-y1)+1;
    }
    glBegin(GL_POINTS);
    glVertex2f(x1+x,y1+y);
    glVertex2f(x1+x,y-y1);
    glVertex2f(x-x1,y1+y);
    glVertex2f(x-x1,y-y1);
    glVertex2f(x+y1,y+x1);
    glVertex2f(x+y1,y-x1);
    glVertex2f(x-y1,y+x1);
    glVertex2f(x-y1,y-x1);
    glEnd();
  }
  glFlush();
}
void delay(long long n)
{
  while(n>0)
  {
    n--;
  }
}

void display()
{

    glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(0.0,0.0,1.0);

      char ch;
      glRasterPos2f(-70,220);
        char msg[]={"Pendulum Clock"};
      for(int i=0;msg[i]!='\0';++i)
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,msg[i]);
        glRasterPos2f(-5,175);
       ch='1';
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
          glRasterPos2f(3,175);
       ch='2';
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
         glRasterPos2f(50,155);
       ch='1';
```

```
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
    glRasterPos2f(70,130);
  ch='2';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(75,90);
  ch='3';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(70,55);
  ch='4';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(40,30);
  ch='5';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-5,15);
  ch='6';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-45,30);
  ch='7';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-75,55);
  ch='8';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
     glRasterPos2f(-85,90);
  ch='9';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-80,130);
  ch='1';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-70,130);
  ch='0';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
   glRasterPos2f(-50,155);
  ch='1';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);
  glRasterPos2f(-40,155);
  ch='1';
   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ch);


 glColor3f(1.0,0.0,0.0);
 midPoint(0,100,100);
 midPoint(0,100,60);
if(angle>225)
 {
   angle=225;
   inc=-inc;
 }
 if(angle<135)
 {
   angle=135;
   inc=-inc;
 }
```

```
      angle+=inc;
      double radian=angle*3.14/180;
      float y2=150*cos((double) radian);
      float x2=150*sin((double) radian);
      midPoint(x2,y2,30);
      glColor3f(0.0,0.0,0.0);
      glLineWidth(2);
      glBegin(GL_LINES);
      {
          glVertex2f(0,0);
          glVertex2f(x2,y2);
          glVertex2f(0,100);
          glVertex2f(0,150);
          glVertex2f(0,100);
          glVertex2f(60,145);

      }
      glEnd();
      glFlush();
      delay(7000000);
      glutPostRedisplay();
}

void keyboard(unsigned char key,int x,int y)
{
  if(key==27)
    exit(0);
}

int main(int argc,char **argv)
{
  glutInit(&argc,argv);
  glutInitWindowSize(500, 500);
  glutCreateWindow("assign8");
  initialize();
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

----------Vechile----------

```
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

float rtri = 0.0f;
float rquad = 0.0f;

void InitGL(int Width,int Height)
{
  glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
  glClearDepth(1.0);
```

```
  glDepthFunc(GL_LESS);
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_SMOOTH);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}

void ReSizeGLScene(int Width, int Height)
{
  if (Height==0)
    Height=1;
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,100.0f);
  glMatrixMode(GL_MODELVIEW);
}

float ballX = -0.4f;
float ballY = 0.0f;
float ballZ = 0.0f;

void drawBall(void)
{
  glColor3f(0.1, 0.1, 0.1);
  glTranslatef(ballX,ballY,ballZ);
  glutSolidSphere(0.2, 20, 20);

   glColor3f(1.0, 1.0, 1.0);
  glTranslatef(ballX+0.4,ballY,ballZ);
  glutSolidSphere(0.1, 20, 20);

   glColor3f(0.1, 0.1, 0.1);
  glTranslatef(ballX+1.5,ballY,ballZ);
  glutSolidSphere(0.2, 20, 20);

  glColor3f(1.0, 1.0, 1.0);
  glTranslatef(ballX+0.4,ballY,ballZ);
  glutSolidSphere(0.1, 20, 20);


}

void DrawGLScene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();

  glBegin(GL_LINES);
  glColor3f(0.0f,0.0f,0.0f);
  glVertex2f(-2.0,0.0);
```

```
 glVertex2f(2.0,0.0);
 glEnd();

 glTranslatef(rtri,0.0f,-6.0f);
 glBegin(GL_POLYGON);
 glColor3f(1.0f,0.3f,0.0f);
 /*glVertex3f(-1.0f, 1.0f, 0.0f);
 glVertex3f(0.4f, 1.0f, 0.0f);
 glVertex3f(1.0f, 0.4f, 0.0f);
 glVertex3f( 1.0f,0.0f, 0.0f);
 glVertex3f(-1.0f,0.0f, 0.0f);*/
 glVertex2f(-1.0,0.0);
 glVertex2f(-1.0,0.5);
 glVertex2f(-0.5,0.5);
 glVertex2f(-0.5,1.0);
 glVertex2f(0.5,1.0);
 glVertex2f(1.0,0.5);
 glVertex2f(1.5,0.5);
 glVertex2f(1.5,0.0);
  glEnd();

  glBegin(GL_POLYGON);
 glColor3f(0.0f,0.2f,0.5f);
 glVertex2f(0.9,0.5);
 glVertex2f(0.4,0.9);
 glVertex2f(0.0,0.9);
 glVertex2f(0.0,0.5);
 glEnd();
  glBegin(GL_POLYGON);
 glColor3f(0.0f,0.2f,0.5f);
 glVertex2f(-0.1,0.9);
 glVertex2f(-0.4,0.9);
 glVertex2f(-0.5,0.5);
 glVertex2f(-0.1,0.5);
 glEnd();

 /*
glBegin(GL_LINES);
 glColor3f(0.0f,0.0f,0.0f);
 glVertex2f(-2.0,0.0);
 glVertex2f(2.0,0.0);
 glEnd();*/


 drawBall();
 rtri+=0.005f;
 if(rtri>2)
     rtri=-2.0f;
 rquad-=15.0f;

// glLoadIdentity();
glLineWidth(10.0);
```

```
  glBegin(GL_LINES);

  glColor3f(0.0f,0.0f,0.0f);
  glVertex2f(-110.0,-0.24);
  glVertex2f(110.0,-0.24);
  glEnd();

  glutSwapBuffers();
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutInitWindowSize(640, 480);
  glutInitWindowPosition(0, 0);
  glutCreateWindow("vehicleLocomotion");
  glutDisplayFunc(&DrawGLScene);
  glutIdleFunc(&DrawGLScene);
  glutReshapeFunc(&ReSizeGLScene);
  InitGL(640, 480);
  glutMainLoop();
  return 0;
}
```