```
#include<iostream>
using namespace std;
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
int xmin=50,ymin=50,xmax=400,ymax=400;
void displayPoint(int x, int y)
{
	glPointSize(2);
	glBegin(GL_POINTS);
	glVertex2i(x, y);
	glEnd();
}
float x01, x02, y01, y02;
int ch;
void SimpleLine(float x1, float y1, float x2, float y2)
{
	float step;

	float dx = x2 - x1;
	float dy = y2 - y1;

if (abs(dx) >= abs(dy))
{
step = abs(dx);
}
else
step = abs(dy);
float Xinc = dx / (float)step;
float Yinc = dy / (float)step;
float x = x1;
float y = y1;

for (int i = 0; i<=step; i++)
{
displayPoint(x, y);
x = x + Xinc;
y = y + Yinc;
}
glFlush();
}
const int L=8, R=4, B=2, T=1;
int x, y, temp; //for four bit code value
int calCode(double x, double y) //to calculate outcode for endpoints of line
{
//p1 =10,40 p2=30 70

int code=0;
if(x>xmax)
code= R;//0010
if(x<xmin) //out code is the unit code given to end pts to end pts of line (above,below,right left) , corners arent checked because all condition are not checked ...only if is used so outcode is added .... if ..else is not used
code= L;
if(y>ymax)
code= T;
if(y<ymin)
code= B;
return(code);
}
void LineClip(double X1, double Y1, double X2, double Y2)
{
unsigned int outcode1 , outcode2;
int accept=0, done=0;
float M= float(Y2-Y1)/(X2-X1); //slope of line
outcode1=calCode(X1,Y1); //To calculate end points outcode value
outcode2=calCode(X2,Y2);
do
```

```
{
if(outcode1==0 & outcode2==0) //completely visible line(inside the window)
{
accept=1;
done=1;
}
else if((outcode1 & outcode2)!=0) //completely invisible line.... single &amp;amp; is for logical anding(completely outside
of window)
{
done=1;
}
else
{
if(outcode1==0)//p1 //if one endpoint is completely inside the window
temp=outcode2; //temp=p2 bit code
else
temp=outcode1;
if(temp & T) //if the point intersects at the top
{
y=ymax;
x= X1 + (ymax-Y1)/M;
}
else if(temp & B ) //if the point intersects at the bottom
{
y= ymin;
x= X1 + (ymin-Y1)/M;
}
else if(temp & L) //if the point intersects at the left
{
x= xmin;
y= Y1 + M*(xmin-X1);
}
else if(temp & R) //if the point intersects at the right
{
x= xmax;
y= Y1+ M*(xmax-X1);
}

if(temp==outcode1)
{
X1= x;
Y1= y;
outcode1=calCode(X1,Y1);
}
if(temp==outcode2)
{
X2= x;
Y2= y;
outcode2=calCode(X2,Y2);
}
}
}while(done==0);
if(accept) //Plot only those points for which accept is equal to 1
{
glClearColor(1.0, 1.0, 1.0, 1.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1, 0, 0);
//To draw Clipping Window
SimpleLine(xmin,ymin,xmax,ymin);
SimpleLine(xmax,ymin,xmax,ymax);
SimpleLine(xmax,ymax,xmin,ymax);
SimpleLine(xmin,ymax,xmin,ymin);
// blue line
glColor3f(0, 0, 1);
SimpleLine(X1,Y1,X2,Y2);
}
}
void keyboard(unsigned char key, int x, int y)
```

```
{
if(key=='c')
{
LineClip(x01,y01,x02,y02);
}
}
void myMouse(int button, int state, int x, int y)
{
static int xst, yst, pt = 0;
if (button == GLUT_LEFT_BUTTON & state == GLUT_DOWN)
{
if (pt == 0)
{
xst = x;
yst = y;
x01 = xst;
y01 = yst;
pt=pt+1;
}
else
{
x02=x;
y02=y;
glColor3f(0, 1, 0);
SimpleLine(xst, yst, x, y);
xst = x;
yst = y;
}
}
else if (button == GLUT_RIGHT_BUTTON & state == GLUT_DOWN)
{
pt=0;
}
//Clear Screen
glFlush();
}
void initialize(void)
{
glClearColor(1.0, 1.0, 1.0, 1.0);
glClear(GL_COLOR_BUFFER_BIT);
// gluOrtho2D(l,r,b,t)
gluOrtho2D(0, 600, 600, 0);
}
void primitives(void)
{
//glClearColor(1.0, 1.0, 1.0, 1.0);
//glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1, 0, 0);
/To draw Clipping Window
SimpleLine(xmin,ymin,xmax,ymin);
SimpleLine(xmax,ymin,xmax,ymax);
SimpleLine(xmax,ymax,xmin,ymax);
SimpleLine(xmin,ymax,xmin,ymin);
glutMouseFunc(myMouse);
glutKeyboardFunc(keyboard);
}
int main(int argc,char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE);
glutInitWindowPosition(0, 0);
glutInitWindowSize(600, 600);
glutCreateWindow("OpenGL - Cohen Sutherland Line Clipping Algo");
initialize();
glutDisplayFunc(primitives);
glutMainLoop();
return 0;
}
```