8)Write a program to implement the Cohen-Sutherland line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping and viewport for displaying the clipped image

```
#include <stdio.h>
#include <GL/glut.h>
#define outcode int
double xmin=50,ymin=50, xmax=100,ymax=100;
double
xvmin=200,yvmin=200,xvmax=300,yvmax=300;
const int RIGHT = 8;
const int LEFT = 2;
const int TOP = 4;
const int BOTTOM = 1;
outcode ComputeOutCode (double x, double y);
void CohenSutherlandLineClipAndDraw (double
x0, double y0,double x1, double y1)
{
outcode outcode0, outcode1, outcodeOut;
bool accept = false, done = false;
outcode0 = ComputeOutCode (x0, y0);
outcode1 = ComputeOutCode (x1, y1);
do{
if (!(outcode0 | outcode1))
{
accept = true;
done = true;
```

```
}
else if (outcode0 & outcode1)
done = true;
else
{
double x, y;
outcodeOut = outcode0? outcode0: outcode1;
if (outcodeOut & TOP)
{
x = x0 + (x1 - x0) * (ymax - y0)/(y1 - y0);
y = ymax;
}
else if (outcodeOut & BOTTOM)
{
x = x0 + (x1 - x0) * (ymin - y0)/(y1 - y0);
y = ymin;
}
else if (outcodeOut & RIGHT)
{
y = y0 + (y1 - y0) * (xmax - x0)/(x1 - x0);
x = xmax;
}
else
{
y = y0 + (y1 - y0) * (xmin - x0)/(x1 - x0);
x = xmin;
}
if (outcodeOut == outcode0)
```

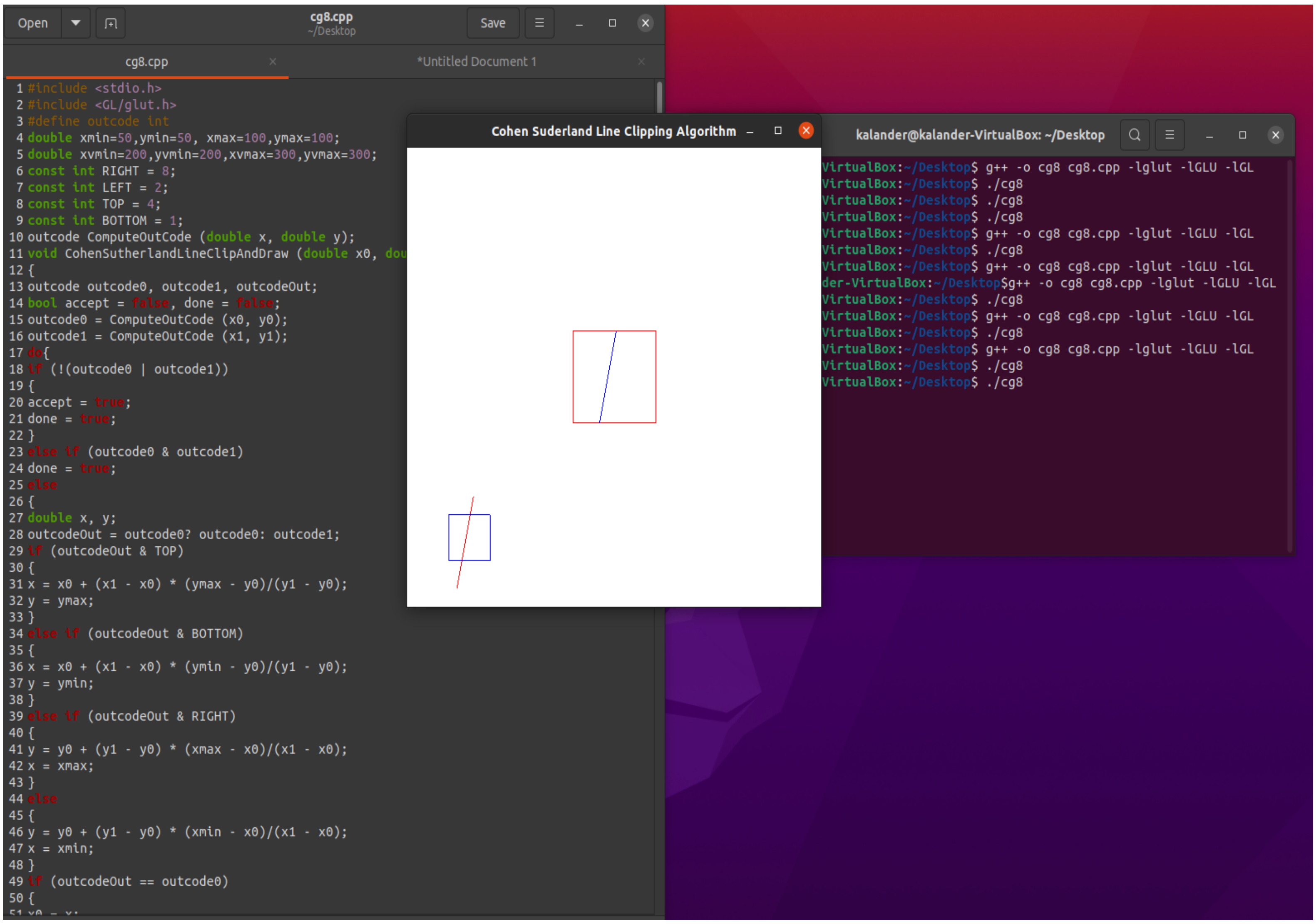
```
{
x0 = x;
y0 = y;
outcode0 = ComputeOutCode (x0, y0);
}
else
{
x1 = x;
y1 = y;
outcode1 = ComputeOutCode (x1, y1);
}
}
}while (!done);
if (accept)
{
double sx=(xvmax-xvmin)/(xmax-xmin);
double sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xvmin, yvmin);
glVertex2f(xvmax, yvmin);
glVertex2f(xvmax, yvmax);
glVertex2f(xvmin, yvmax);
glEnd();
```

```
glColor3f(0.0,0.0,1.0);
glBegin(GL_LINES);
glVertex2d (vx0, vy0);
glVertex2d (vx1, vy1);
glEnd();
}
}
outcode ComputeOutCode (double x, double y)
{
outcode code = 0;
if (y > ymax)
code |= TOP;
else if (y < ymin)
code |= BOTTOM;
if (x > xmax)
code |= RIGHT;
else if (x < xmin)
code |= LEFT;
return code;
}
void display()
{
double x0=60,y0=20,x1=80,y1=120;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2d (x0, y0);
glVertex2d (x1, y1);
```

```
glEnd();
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
 glVertex2f(xmin, ymin);
 glVertex2f(xmax, ymin);
 glVertex2f(xmax, ymax);
 glVertex2f(xmin, ymax);
glEnd();
CohenSutherlandLineClipAndDraw(x0,y0,x1,y1);
glFlush();
}
void myinit()
{
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,499.0,0.0,499.0);
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Cohen Suderland Line
Clipping Algorithm");
```

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

Output:



Open
cg8.cpp
~/Desktop
Save
cg8.cpp
*Untitled Document 1
Cohen Suderland Line Clipping Algorithm
kalander@kalander-VirtualBox: ~/Desktop

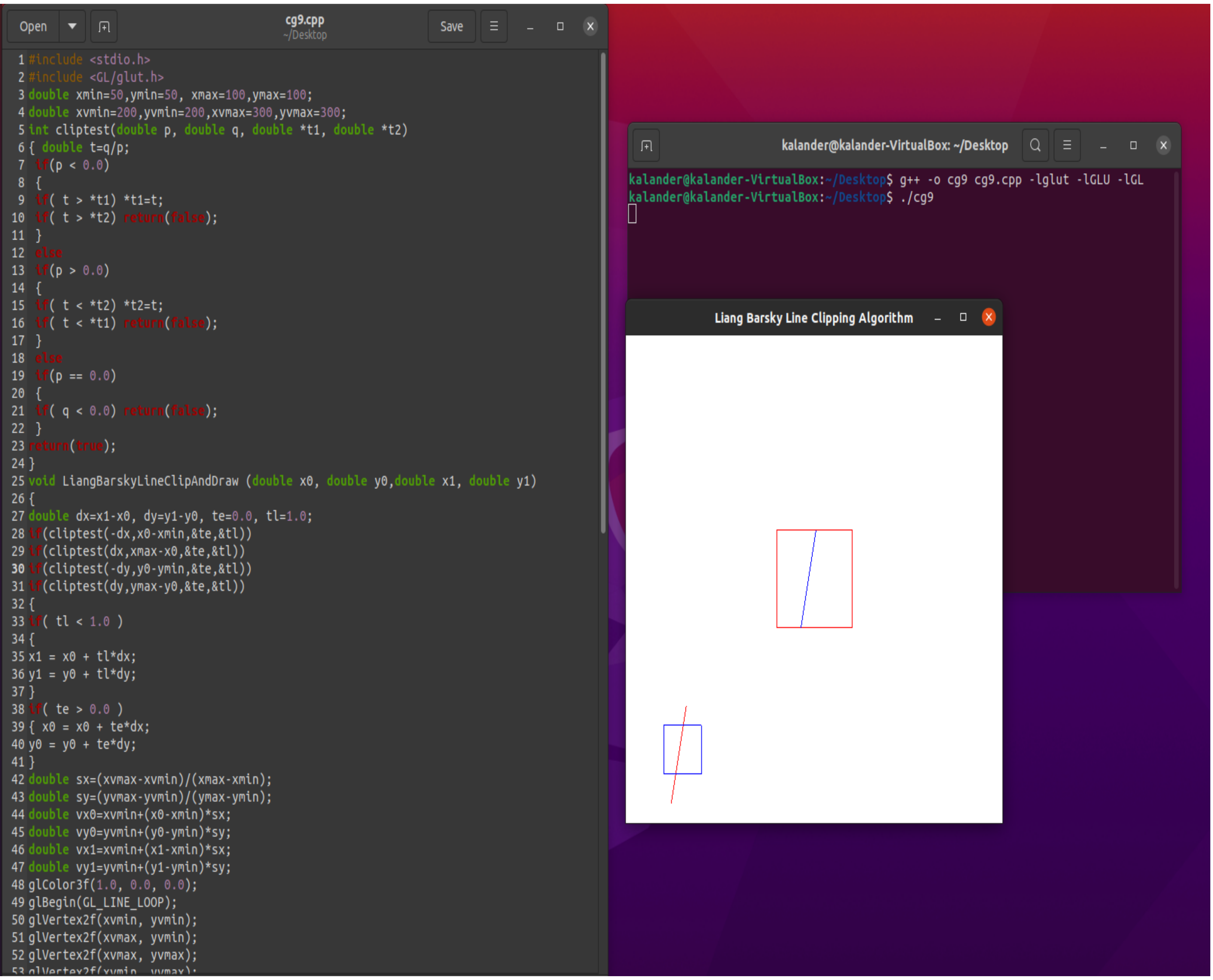
```
#include <stdio.h>
#include <GL/glut.h>
#define outcode int
double xmin=50,ymin=50, xmax=100,ymax=100;
double xvmin=200,yvmin=200,xvmax=300,yvmax=300;
const int RIGHT = 8;
const int LEFT = 2;
const int TOP = 4;
const int BOTTOM = 1;
outcode ComputeOutCode (double x, double y);
void CohenSutherlandLineClipAndDraw (double x0, dou
{
outcode outcode0, outcode1, outcodeOut;
bool accept = false, done = false;
outcode0 = ComputeOutCode (x0, y0);
outcode1 = ComputeOutCode (x1, y1);
do{
if (!(outcode0 | outcode1))
{
accept = true;
done = true;
}
else if (outcode0 & outcode1)
done = true;
else
{
double x, y;
outcodeOut = outcode0? outcode0: outcode1;
if (outcodeOut & TOP)
{
x = x0 + (x1 - x0) * (ymax - y0)/(y1 - y0);
y = ymax;
}
else if (outcodeOut & BOTTOM)
{
x = x0 + (x1 - x0) * (ymin - y0)/(y1 - y0);
y = ymin;
}
else if (outcodeOut & RIGHT)
{
y = y0 + (y1 - y0) * (xmax - x0)/(x1 - x0);
x = xmax;
}
else
{
y = y0 + (y1 - y0) * (xmin - x0)/(x1 - x0);
x = xmin;
}
if (outcodeOut == outcode0)
{
```

```
VirtualBox:~/Desktop$ g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
der-VirtualBox:~/Desktop$g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ g++ -o cg8 cg8.cpp -lglut -lGLU -lGL
VirtualBox:~/Desktop$ ./cg8
VirtualBox:~/Desktop$ ./cg8
```

9) Write a program to implement the Liang-Barsky line clipping algorithm. Make provision to specify the input for multiple lines, window for clipping and viewport for displaying the clipped image.

```
#include <stdio.h>
#include <GL/glut.h>
double xmin=50,ymin=50, xmax=100,ymax=100;
double
xvmin=200,yvmin=200,xvmax=300,yvmax=300;
int cliptest(double p, double q, double *t1,
double *t2)
{ double t=q/p;
 if(p < 0.0)
 {
 if( t > *t1) *t1=t;
 if( t > *t2) return(false);
 }
 else
 if(p > 0.0)
 {
 if( t < *t2) *t2=t;
 if( t < *t1) return(false);
 }
 else
 if(p == 0.0)
 {
 if( q < 0.0) return(false);
 }
return(true);
}
void LiangBarskyLineClipAndDraw (double x0,
double y0,double x1, double y1)
{
double dx=x1-x0, dy=y1-y0, te=0.0, tl=1.0;
if(cliptest(-dx,x0-xmin,&te,&tl))
if(cliptest(dx,xmax-x0,&te,&tl))
if(cliptest(-dy,y0-ymin,&te,&tl))
if(cliptest(dy,ymax-y0,&te,&tl))
{
if( tl < 1.0 )
{
x1 = x0 + tl*dx;
y1 = y0 + tl*dy;
}
if( te > 0.0 )
{ x0 = x0 + te*dx;
y0 = y0 + te*dy;
}
double sx=(xvmax-xvmin)/(xmax-xmin);
double sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xvmin, yvmin);
glVertex2f(xvmax, yvmin);
glVertex2f(xvmax, yvmax);
glVertex2f(xvmin, yvmax);
glEnd();
glColor3f(0.0,0.0,1.0);
glBegin(GL_LINES);
glVertex2d (vx0, vy0);
glVertex2d (vx1, vy1);
glEnd();
}
}
void display()
{
double x0=60,y0=20,x1=80,y1=120;
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2d (x0, y0);
glVertex2d (x1, y1);
glEnd();
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
 glVertex2f(xmin, ymin);
 glVertex2f(xmax, ymin);
 glVertex2f(xmax, ymax);
 glVertex2f(xmin, ymax);
glEnd();
LiangBarskyLineClipAndDraw(x0,y0,x1,y1);
glFlush();
}
void myinit()
{
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,499.0,0.0,499.0);
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Liang Barsky Line Clipping
Algorithm");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

Output:


Open
Save
cg9.cpp
~/Desktop
kalander@kalander-VirtualBox: ~/Desktop
Liang Barsky Line Clipping Algorithm

```
#include <stdio.h>
#include <GL/glut.h>
double xmin=50,ymin=50, xmax=100,ymax=100;
double xvmin=200,yvmin=200,xvmax=300,yvmax=300;
int cliptest(double p, double q, double *t1, double *t2)
{ double t=q/p;
 if(p < 0.0)
 {
 if( t > *t1) *t1=t;
 if( t > *t2) return(false);
 }
 else
 if(p > 0.0)
 {
 if( t < *t2) *t2=t;
 if( t < *t1) return(false);
 }
 else
 if(p == 0.0)
 {
 if( q < 0.0) return(false);
 }
return(true);
}
void LiangBarskyLineClipAndDraw (double x0, double y0,double x1, double y1)
{
double dx=x1-x0, dy=y1-y0, te=0.0, tl=1.0;
if(cliptest(-dx,x0-xmin,&te,&tl))
if(cliptest(dx,xmax-x0,&te,&tl))
if(cliptest(-dy,y0-ymin,&te,&tl))
if(cliptest(dy,ymax-y0,&te,&tl))
{
if( tl < 1.0 )
{
x1 = x0 + tl*dx;
y1 = y0 + tl*dy;
}
if( te > 0.0 )
{ x0 = x0 + te*dx;
y0 = y0 + te*dy;
}
double sx=(xvmax-xvmin)/(xmax-xmin);
double sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_LOOP);
glVertex2f(xvmin, yvmin);
glVertex2f(xvmax, yvmin);
glVertex2f(xvmax, yvmax);
glVertex2f(xvmin, yvmax);
```

```
kalander@kalander-VirtualBox:~/Desktop$ g++ -o cg9 cg9.cpp -lglut -lGLU -lGL
kalander@kalander-VirtualBox:~/Desktop$ ./cg9
```