8. Solve the following

a. Write a program to implement Cohen-Sutherland clipping.

b. Write a program to implement Liang - Barsky Line Clipping Algorithm.

a. Write a program to implement Cohen-Sutherland clipping.

Solution:-

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
float slope;
int x,y,x1,y1,i, xc,yc;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
```

```
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm **********");
printf("\n Now, enter XMin, YMin =");

scanf("%d %d",&W_xmin,&W_ymin);
printf("\n First enter XMax, YMax =");
scanf("%d %d",&W_xmax,&W_ymax);
printf("\n Please enter intial point x and y= ");
scanf("%d %d",&x,&y);
printf("\n Now, enter final point x1 and y1= ");
scanf("%d %d",&x1,&y1);
cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax)  {
rcode_begin[0]=1;     // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;          // Bottom
```

```
flag=1;
}
if(x>W_xmax)  {
rcode_begin[2]=1;          // Right
flag=1;
}
if(x<W_xmin)   {
rcode_begin[3]=1;          //Left
flag=1;
}

//end point of Line
if(y1>W_ymax){
rcode_end[0]=1;          // Top
flag=1;
}
if(y1<W_ymin) {
rcode_end[1]=1;          // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;          // Right
```

```
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;          //Left
flag=1;
 }
if(flag==0)
{
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
 flag=0;
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
else{
slope=(float)(y1-y)/(x1-x);
```

```
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)      // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)     // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;

}
if(rcode_begin[0]==0 && rcode_begin[1]==1)    // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;
```

```
}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)   //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)      // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)     // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;

}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
```

```
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
}
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

b. Write a program to implement Liang - Barsky Line Clipping Algorithm.

Solution:-

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

```
#include<graphics.h>
#include<math.h>
#include<dos.h>

void main()
{
int i,gd=DETECT,gm;
int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
float t1,t2,p[4],q[4],temp;
x1=120;
y1=120;
x2=300;
y2=300;
xmin=100;
ymin=100;
xmax=250;
ymax=250;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
rectangle(xmin,ymin,xmax,ymax);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
```

```
p[1]=dx;
p[2]=-dy;
p[3]=dy;
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;
for(i=0;i<4;i++)
{
if(p[i]==0)
{
printf("line is parallel to one of the clipping boundary");
if(q[i]>=0)
{
if(i<2)
{
if(y1<ymin)
{
y1=ymin;
}
if(y2>ymax)
{
```

```
y2=ymax;
}
line(x1,y1,x2,y2);
}
if(i>1)
{
if(x1<xmin)
{
x1=xmin;
}
if(x2>xmax)
{
x2=xmax;
}
line(x1,y1,x2,y2);
}
}
}
}
t1=0;
t2=1;
for(i=0;i<4;i++)
```

```
{
temp=q[i]/p[i];
if(p[i]<0)
{
if(t1<=temp)
t1=temp;
}
else
{
if(t2>temp)
t2=temp;
}
}
if(t1<t2)
{
xx1 = x1 + t1 * p[1];
xx2 = x1 + t2 * p[1];
yy1 = y1 + t1 * p[3];
yy2 = y1 + t2 * p[3];
line(xx1,yy1,xx2,yy2);
}
delay(5000);
```

```
closegraph();
}
```