}

P2. Algorithm :-

Step 1 : Start

Step 2 : Read 2 end points of line as P1 (x1, y1) and P2 (x2, y2)

Step 3 : Read 2 corner points of the clipping window (left-top and right-bottom) as (wx1, wy1) and (wx2, wy2)

Step 4 : Assign the region codes for 2 endpoints p1 and p2 using following steps :-

Initialize Code with 0000

Set bit 1 if $x < wx1$

Set bit 2 if $x > wx2$

Set bit 3 if $y < wy2$

Set bit 4 if $y > wy1$

Step 5 :- Check for visibility of line
a) if region codes for both endpoints are zero then line is completely visible. Draw the line go to step 10.
b) if region codes for endpoints and not zero and logical ANDing of them is also non zero then line is invisible. Discard the line & move to step 10.
c) if it does not satisfy 5a & 5b then line is partially visible.

Step 6 :- Determine the intersecting edge of clipping window as follows :-
a) if region codes for both endpoints are non zero find intersection points P1' and P2' with boundary edges.
b) if region codes for any one end point is non zero then find intersection point P1' or P2'.

Step 7 :- Divide the line segments Considering intersection points.

Step 8 :- Reject line segment if any end point of line appears outside of any boundary.

Step 9 : Draw the clipped line segment.

Step 10 : Stop.

Program :

```
# 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, &y1);
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;
flag = 1;
}
if(y < w_ymin) {
rcode_begin[1] = 1;
flag = 1;
}
if(x > w_xmax) {
rcode_begin[2] = 1;
flag = 1;
}
if(x < w_xmin) {
rcode_begin[3] = 1;
flag = 1;
}
if(y1 > w_ymax) {
rcode_end[0] = 1;
flag = 1;
}
if(y1 < w_ymin) {
rcode_end[1] = 1;
flag = 1;
```

```
}
if (x1<w_xmin) {
rcode_end[3]=1;
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)
{
 else {
slope=(float)(y2-y)/(x2-x);
if(rcode_begin[2]==0 && rcode_
                begin[3]==1)
{
y=y+(float)(w_xmin-x)*slope;
 x=w_xmin;
}
if(rcode_begin[2]==1 && rcode_begin
                [3]==0)
{
```

```
y = y + (float) (w - xmax - x) * slope ;
x = w - xmax ;
}
if (rcode_begin [0] == 1 && rcode_begin
                [1] == 0)
{
x = x + (float) (w - ymax - y) / slope ;
y = w - ymax ;
}
if (rcode_begin[0] == 0 && rcode_begin
                [1] == 1)
{
x = x + (float) (w - ymin - y) / slope ;
y = w - ymin ;
}
if (rcode_end [2] == 0 && rcode_end [3]
                == 1)
{
y1 = y1 + (float) (w - xmin - x1) * slope ;
x1 = w - xmin ;

}
if (rcode_end[2] == 1 && rcode_end[3]
                == 0)
{
y1 = y1 + (float) (w - xmax - x1) * slope ;
x1 = w - xmax ;
}
if (rcode_end [0] == 1 && rcode_end [1] == 0)
{
```

```
x1=x1+(float)(w-ymax-y1)/slope;
y1=w-ymax;
}
if(rcode-end[0]==0 && rcode-end[1]==1)
{
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();
}
```