Name: Shawn Louis
SE Computer
Batch B
Roll No. 31

EXPERIMENT 08	
Title	**IMPLEMENTATION OF SUTHERLAND – HODGEMAN POLYGON CLIPPING ALGORITHM**
Objective	To write a C program to implement Sutherland – Hodgeman Polygon clipping algorithm.
Algorithm	1. Start 2. Get the minimum and maximum coordinates of both window and view port. 3. Get the number of sides of a polygon and its corresponding coordinates. 4. If the polygon lies within region code window, display it. 5. If any one of the polygon side is neither inside nor outside the boundary, find the point of intersection and clip the regions that lies outside the boundary. 6. Display the polygon after clipping 7. Stop

Program	#include<stdio.h> #include<graphics.h> #include<math.h> typedef struct { float x; float y;

```
}PT;

int n;

main()

{

int i,j,gd,gm;

PT d,p1,p2,p[20],pi1,pi2,pp[20];

detectgraph(&gd,&gm);

initgraph(&gd,&gm," c:\\tc\\bgi ");

/* Read coordinates of clipping window */

printf("Enter coordinates (left,top) of point1 : ");

scanf("%f %f",&p1.x,&p1.y);

printf("Enter coordinates (right,bottom) of point2 : ");

scanf("%f %f",&p2.x,&p2.y);

/* Enter the number of vertex */

printf("Enter the number of vertex : ");

scanf("%d",&n);

/* Read vertex coordinates of clipping window */

for(i=0;i<n;i++)
```

```
{
printf("Enter coordinates of vertex%d : ",i+1);
scanf("%f %f",&p[i].x,&p[i].y);
}
p[i].x = p[0].x;
p[i].y = p[0].y;
cleardevice();
drawpolygon(p,n);
rectangle(p1.x,p1.y,p2.x,p2.y);
getch();
left(p1,p,pp);
right(p2,p,pp);
top(p1,p,pp);
bottom(p2,p,pp);
cleardevice();
rectangle(p1.x,p1.y,p2.x,p2.y);
drawpolygon(p,n);
getch();
```

```
closegraph();

}

left(PT p1,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].x < p1.x && p[i+1].x >= p1.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)* (p1.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p1.x;
```

```
j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;

}

if(p[i].x > p1.x && p[i+1].x >= p1.x)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].x > p1.x && p[i+1].x <= p1.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)* (p1.x-p[i].x)+p[i].y;

}

else
```

```
{
pp[j].y = p[i].y;
}
pp[j].x = p1.x;
j++;
}
}
for(i=0;i<j;i++)
{
p[i].x = pp[i].x;
p[i].y = pp[i].y;
}
p[i].x = pp[0].x;
p[i].y = pp[0].y;
n=j;
}
right(PT p2,PT p[20],PT pp[20])
{
```

```
int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].x > p2.x && p[i+1].x <= p2.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)* (p2.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p2.x;

j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;
```

```
}

if(p[i].x < p2.x && p[i+1].x <= p2.x)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].x < p2.x && p[i+1].x >= p2.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)* (p2.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p2.x;
```

```
j++;

}

}

for(i=0;i<j;i++)

{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

top(PT p1,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].y < p1.y && p[i+1].y >= p1.y)
```

```
{
if(p[i+1].y-p[i].y!=0)
{
pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)* (p1.y-p[i].y)+p[i].x;
}
else
{
pp[j].x = p[i].x;
}
pp[j].y = p1.y;
j++;
pp[j].x=p[i+1].x;
pp[j].y=p[i+1].y;
j++;
}
if(p[i].y > p1.y && p[i+1].y >= p1.y)
{
pp[j].y = p[i+1].y;
```

```
pp[j].x = p[i+1].x;

j++;

}

if(p[i].y > p1.y && p[i+1].y <= p1.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)* (p1.y-p[i].y)+p[i].x;

}

else

{

pp[j].x = p[i].x;

}

pp[j].y = p1.y;

j++;

}

}

for(i=0;i<j;i++)
```

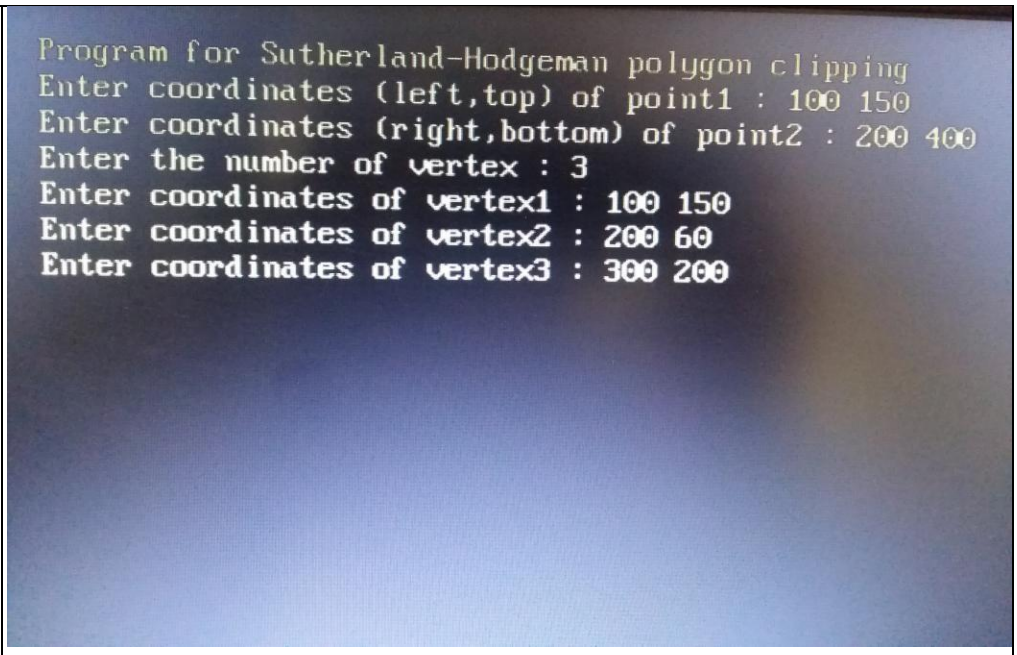
```
{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

bottom(PT p2,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].y > p2.y && p[i+1].y <= p2.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)* (p2.y-p[i].y)+p[i].x;
```

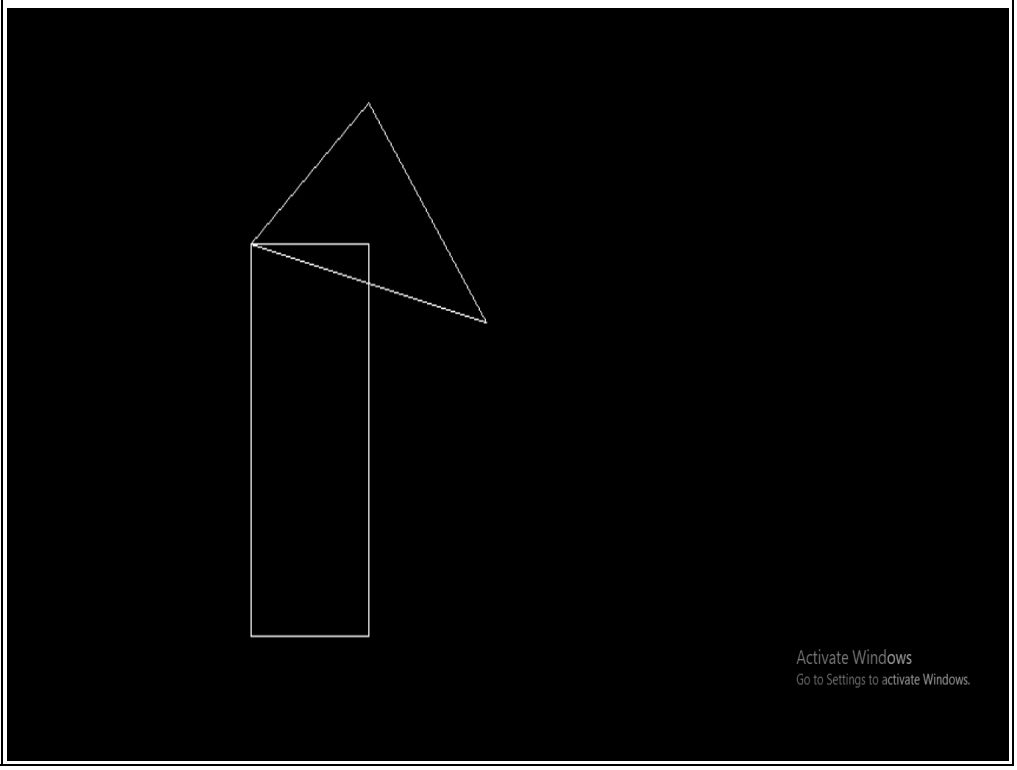
```
}
else
{
pp[j].x = p[i].x;
}
pp[j].y = p2.y;
j++;
pp[j].x=p[i+1].x;
pp[j].y=p[i+1].y;
j++;
}
if(p[i].y < p2.y && p[i+1].y <= p2.y)
{
pp[j].y = p[i+1].y;
pp[j].x = p[i+1].x;
j++;
}
if(p[i].y < p2.y && p[i+1].y >= p2.y)
```

```
{
if(p[i+1].y-p[i].y!=0)
{
pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)* (p2.y-p[i].y)+p[i].x;
}
else
{
pp[j].x = p[i].x;
}
pp[j].y = p2.y;
j++;
}
}
for(i=0;i<j;i++)
{
p[i].x = pp[i].x;
p[i].y = pp[i].y;
}
```

	p[i].x = pp[0].x;
	
	p[i].y = pp[0].y;
	
	n=j;
	
	}
	
	drawpolygon(PT x[20],int n)
	
	{
	
	int i;
	
	for(i=0;i<n-1;i++)
	
	{
	
	line(x[i].x,x[i].y,x[i+1].x,x[i+1].y);
	
	}
	
	line(x[i].x,x[i].y,x[0].x,x[0].y);
	
	}

Output	

<table>
<tr>
<td></td>
<td>


</td>
</tr>
<tr>
<td></td>
<td>


</td>
</tr>
</table>

Conclusion	**Thus a C program to implement Sutherland – Hodgeman Polygon clipping algorithm was written and executed.**