```
#include<iostream.h>
#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
#include<dos.h>

struct edge

{
    int x1,y1,x2,y2,flag;
};



int main()
{
    int n,i,j,k,gd,gm,x[10],y[10],ymax=0,ymin=480,yy,temp;
    struct edge ed[10],temped;  //ed[3].x1,ed[3].y1
    float dx,dy,m[10],x_int[10],inter_x[10];
   initgraph(&gd,&gm,"c://Turboc3//BGI");

     cout<<"\n Enter the number of vertices of the graph: "; cin>>n;
     cout<<"\n Enter the vertices: \n";
    for(i=0;i<n;i++)
    {
     cout<<"x"<<i<<":"; cin>>x[i];
     cout<<"y"<<i<<":"; cin>>y[i];
     if(y[i]>ymax)
         ymax=y[i];
     if(y[i]<ymin)
         ymin=y[i];
     ed[i].x1=x[i];   //ed[0].x1=x[0] ed[0].y1=y[0];
     ed[i].y1=y[i];

    }
```

```
for(i=0;i<n-1;i++)  //store the edge information
{
 ed[i].x2=ed[i+1].x1;  //ed[0].x2=ed[1].x1;
 ed[i].y2=ed[i+1].y1;
 ed[i].flag=0;
}
 ed[i].x2=ed[0].x1;   //i=n-1
 ed[i].y2=ed[0].y1;
    ed[i].flag=0;

for(i=0;i<n-1;i++)  //check for y1>y2 if not interchange it
{
 if(ed[i].y1<ed[i].y2)
 {
 temp=ed[i].x1;
 ed[i].x1=ed[i].x2;
 ed[i].x2=temp;
 temp=ed[i].y1;
 ed[i].y1=ed[i].y2;
 ed[i].y2=temp;
 }
}
/*for(i=0;i<n;i++)  //draw polygon
{
 line(ed[i].x1,ed[i].y1,ed[i].x2,ed[i].y2);
} */

for(i=0;i<n-1;i++)  //storing the edges as y1,y2,x1
{
    for(j=0;j<n-1;j++)
    {
        if(ed[j].y1<ed[j+1].y1)
        {
        temped=ed[j];
        ed[j]=ed[j+1];
        ed[j+1]=temped;
        }
        if(ed[j].y1==ed[j+1].y1)
        {
            if(ed[j].y2<ed[j+1].y2)
            {
                temped=ed[j];
                ed[j]=ed[j+1];
                ed[j+1]=temped;
            }
            if(ed[j].y2==ed[j+1].y2)
            {
                if(ed[j].x1<ed[j+1].x1)
                {
                temped=ed[j];
                ed[j]=ed[j+1];
                ed[j+1]=temped;
                }
            }
        }
    }
}

for(i=0;i<n;i++)  //calculate 1/slope
{
    dx=ed[i].x2-ed[i].x1;
    dy=ed[i].y2-ed[i].y1;
    if(dy==0)
        m[i]=0;
    else
        m[i]=dx/dy;
    inter_x[i]=ed[i].x1;

}
yy=ymax;
```

```
while(yy>ymin)  //Mark active edges
{
    for(i=0;i<n;i++)
    {
        if(yy>ed[i].y2 && yy<=ed[i].y1 && ed[i].y1!=ed[i].y2)
            ed[i].flag=1;
        else
            ed[i].flag=0;
    }



j=0;
for(i=0;i<n;i++)  //Finding x intersections
{
 if(ed[i].flag==1)
 {
     if(yy==ed[i].y1)
     {
      x_int[j]=ed[i].x1;
      j++;
      /*if(ed[i-1].y1==yy&&ed[i-1].y1<yy)
      {
            x_int[j]=ed[i].x1;
            j++;
      }
      if(ed[i+1].y1==yy&&ed[i+1].y1<yy)
      {
            x_int[j]=ed[i].x1;
            j++;
      } */

     }

     else
     {
          x_int[j]=inter_x[i]+(-m[i]);
                inter_x[i]=x_int[j];
                j++;
           }
      }
 }

 for(i=0;i<j;i++)  //sorting the x intersections
 {
      for(k=0;k<j-1;k++)
      {
           if(x_int[k]>x_int[k+1])
           {
           temp=x_int[k];
           x_int[k]=x_int[k+1];
           x_int[k+1]=temp;
           }
      }
 }

 for(i=0;i<j;i+=2)  //Extracting x values to draw a line
 {
  line(x_int[i],yy,x_int[i+1],yy);
  delay(100);
 }
 yy--;
}  //end of while loop
 delay(3000);
 getch();
 closegraph();
 return 0;

}
```