**Java Solution code with explanation:**

import java.util.Scanner;

class Q3

{

public static void main(String args[])

{

double l=0,v[][],x,y,l2=0;

//(x,y) represent the mid point of the considered edge

//v[][] stores the points in x,y pairs

int i,j;//used to loop

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number of initial edges:");

int e=sc.nextInt();

v=new double[e+1][2];

for(i=0;i<e+1;i++)

{

v[i][0]=sc.nextDouble();//x co-ordinate

v[i][1]=sc.nextDouble();//y co-ordinate

}

int n=sc.nextInt();

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

{

l+=Math.sqrt((v[i][0]-v[i+1][0])\*(v[i][0]-v[i+1][0])+(v[i][1]-v[i+1][1])\*(v[i][1]-v[i+1][1]));

//summing distances of the initial edges

}

n-=e;//reducing total edges to be considered by the initial count

int counter = 0;

for(i=0;i<n;i++) // iterating through all the edges remaining and generating them as needed

{

//mid-point theorem: probably everyone did it in 8th :)

x=(v[0][0]+v[1][0])/2;

y=(v[0][1]+v[1][1])/2;

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

{

v[j][0]=v[j+1][0];

v[j][1]=v[j+1][1];

//shifting the vertices behind by one and deleting an edge

// if vertices a,b,c,d are stored then they correspond to edges ab bc and cd

//(QUE CAN ALSO BE USED)

}

v[e][0]=x;//assigning new verices to the top

v[e][1]=y;

l+=Math.sqrt((v[e][0]-v[e-1][0])\*(v[e][0]-v[e-1][0])+(v[e][1]-v[e-1][1])\*(v[e][1]-v[e-1][1]));//adding new distance

/\*for the second part of the challenge:

\*if the next e edges generated are so small that they do not cause any significant increase in the length then

\*they can be ignored and the loop can be terminated. The reason for taking next e edges becomes evident when a

\*figure which is having much greater length than width is considered. Edges along the length will take longer

\*to become negligible.

\*/

if(l == l2)

{

counter++;

if(counter>e)

break;

else

counter = 0;

}

else

l2 = l;

}

System.out.println(l);//output

}

}