Assignment 1

The Problem:

Given a bunch of polygons, determine if they are all well-formed, and if so, whether they form an interconnected region. Find the "shortest" path to each polygon from the initial polygon.

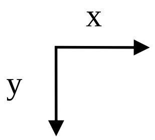
Input file:

First line contains the total number of polygons in the file, say N.

Each following line describes a polygon. Polygon number k will be described in (k+1)th line. The first polygon is called the ***initial polygon***.

A polygon is written as a sequence of non-negative integers:

x1 y1 x2 y2 x3 y3 …. xn yn



n is the number of vertices of the polygon. Starting from any vertex (x1,y1) of the polygon, the next sequence (x2,y2) … (xn,yn) must be of the vertices encountered as we go counter-clockwise around the polygon starting from (x1,y1).

- The numbers in the input file are separated by a single space.
- The maximum number of vertices is 20, i.e. each row will contain at most 40 non-negative integers.

Example input file:

```
4

0 0 10 10 20 10 10 0

10 0 10 20 15 20 15 0

20 0 20 10 30 20

0 0 10 10 10 0 0 10
```

BAD polygons

A polygon that is not well formed is called a bad polygon. This can happen, for example, if the format described above is not followed in the input file in terms of numbers and spaces, when there are non-integers or negative integers in the line, when there are an odd number of non-negative integers in a line, or if the number of non-negative integers exceeds 40.

Two more cases of bad polygons needs to be checked w.r.t the sequence (x1,y1) … (xn,yn):

- The sequence of line segments connecting (x1,y1) to (x2,y2), (x2,y2) to (x3,y3), …, (xn,yn) to (x1,y1) need to be non-intersecting, i.e. the boundary of the polygon should not cross itself.

- The sequence is anti-clockwise i.e. the interior of the polygon is towards the left as we traverse the boundary and not towards the right.

If you encounter a bad polygon in the input file, print the single word BAD in the first line of the output file followed by a space followed by the number of the polygon.

Example:

For the input file that we used above, we should print in the output file:

BAD 4

Overlapping polygons

If two polygons intersect with non-zero intersection area, then they are said to ***overlap***.

Note that if two polygons touch each other at a vertex or edge, then this does not mean that they overlap.

In the input file above, polygons 1 and 2 overlap, but polygons 2 and 3 do not overlap and polygons 3 and 1 do not overlap.

A graph can be created with nodes as polygons. An edge is drawn between two polygons iff they overlap.

The given set of polygons is called ***disconnected*** if the graph is disconnected. When you encounter such an input, print in the output file the single word DISCONNECTED followed by a space followed by the number of the **first** polygon that cannot be reached from the **initial polygon**.

For example, if the input file is

3

0 0 10 10 20 10 10 0

10 0 10 20 15 20 15 0

20 0 20 10 30 20

Then the output file should read:

DISCONNECTED 3

Distance between polygons

If polygon Pi overlaps with Pj, then a directed edge can be drawn from Pi to Pj carrying weight w given by:

w(Pi,Pj) = distance between centroid of Pi and Pj,

where centroid of a polygon with points (x1,y1) … (xn,yn) is (x',y') with x' = 1/n (Σ xi) and y' = 1/n (Σ yi)

w is a rudimentary measure of the distance between overlapping polygons. Note that:

- w(Pi,Pi) = 0
- If the polygons don't overlap, then the distance is infinity (by definition).

A more comprehensive measure of distance is obtained by finding the weight of the path from one polygon to another through a sequence of overlapping polygons.

The ***distance*** between polygon Pi and Pj, distance(Pi,Pj) is defined as the length of the shortest path from Pi to Pj in the graph whose edges are weighted by w.

If the given input file does not have BAD polygons and is not DISCONNECTED, then the output file should contain the distance to each polygon from the initial polygon. The first line should contain the word CONNECTED and line number k should contain the distance to polygon k rounded to 2 decimal places. Note that we omit the initial polygon, since the distance to it is known to be 0.

If the input file is:

```
4
0 0 0 20 20 20 20 0
15 15 15 30 30 30 30 15
15 0 15 5 30 5 30 0
25 0 25 20 30 20 30 0
```

Then the output file will be:

```
CONNECTED
17.68
14.58
23.59
```

Submissions

Each submission must contain:

- The program written in a single file, named as your_roll_no.ext, where the extension ext can be either c, cpp, pas, py or java.
- Three test cases with each test case having an input file and an output file. There should be one input file which has a BAD polygon, one that is DISCONNECTED and one that is CONNECTED. Input files should be named your_roll_no-input1.in, your_roll_no-input1.out, ...
- A written document in PDF format (max 4 pages) containing a short description of your algorithm, the test cases you used to test the algorithm and analysis of the complexity of your algorithm.
- The program and test cases are to be submitted into the IARCS Contest and Moodle Servers, respectively (described below), while the document is to be submitted via a private posting to instructors on Piazza.

Testing your program on the server

Uploading the program

Your source code needs to be uploaded on the IARCS Contest Server https://opc.iarcs.org.in:7777 to the task **Polygon.** Ignore the task **PolygonReference**. Ensure that the code is a single file, in any of the supported languages given on the server. The language is determined by the file name extension you provide.

Your uploaded code will be compiled and tested first against a **fixed set of test cases** for validation. The server will notify you in case compilation fails, the program encounters a seg-fault, or the time/memory limit is exceeded for the program.

If your code passes the fixed test cases, it will then be run against the global set of test cases (that includes those submitted by every student) after a fixed interval. The results of these runs will **not** be displayed to you directly.

Uploading the test cases

Your test cases (input and output files) should be uploaded on the IARCS Moodle Server http://moodle.iarcs.org.in as a single zip file.

Your test cases will be validated using a reference implementation that checks the syntax of the input files and the accuracy of the corresponding output files. Note that the validity of a test case depends on both – the input and the output file. They should be in the correct format and the output should be correct with respect to the input.

After validation, the submitted test cases will be added to the IARCS Contest Server and used to check the code of every other student as well. The quality of the test cases also carries some weight in the final score, so you are encouraged to come up with corner cases that are difficult to pass.

Uploading the report

The report is to be submitted separately on Piazza, as a private post to the instructors in PDF format.

The Contest and Results

Marks will be given based on:

- the number of the **test cases** submitted by your colleagues that your program is able to pass successfully
- number of programs that cannot pass your submitted test cases, as a measure of the quality of the test cases