Concave Polygon ★

Points: 553.1800000000001 Rank: 1150

Problem Submissions Leaderboard

RATE THIS CHALLENGE

★ ★ ★ ★ ★

You are given the cartesian coordinates of a set of points in a $2D$ plane (in no particular order). Each of these points is a corner point of some Polygon, P, which is not self-intersecting in nature. Can you determine whether or not P is a concave polygon?

Input Format

The first line contains an integer, N, denoting the number of points.
The N subsequent lines each contain 2 space-separated integers denoting the respective x and y coordinates of a point.

Constraints

- $3 \leq N \leq 1000$
- $0 \leq x, y \leq 1000$

Output Format

Print `YES` if P is a concave polygon; otherwise, print `NO`.

Sample Input

```
4
0 0
0 1
1 1
1 0
```

Sample Output

```
NO
```

Explanation

The given polygon is a square, and each of its 4 internal angles are 90°. As none of these are over 180°, the polygon is not concave and we print `NO`.

Scoring

The percentage score awarded for your submission will be:

```
100 - 2*(percentage of tests which you solve incorrectly)
```

If this value is negative, the percentage score for your submission will be 0.
So if you get half or more of the tests incorrect, your score will be a zero.

Change Theme Language Haskell

```haskell
import Control.Monad
import Data.List (sortBy)

dotProduct :: (Double, Double) -> (Double, Double) -> (Double, Double) -> Double
dotProduct (x0, y0) (x1, y1) (x2, y2) = let (x1', y1') = (x1-x0, y1-y0)
                                            (x2', y2') = (x2-x0, y2-y0)
                                        in x1'*x2' + y1'*y2'


crossProduct :: (Double, Double) -> (Double, Double) -> (Double, Double) -> Double
crossProduct (x0, y0) (x1, y1) (x2, y2) = let (x1', y1') = (x1-x0, y1-y0)
                                              (x2', y2') = (x2-x0, y2-y0)
                                          in x1'*y2' - y1'*x2'

getAngle :: (Double, Double) -> (Double, Double) -> (Double, Double) -> Double
getAngle z a b = let d = dotProduct z a b
                     c = crossProduct z a b
                     angle = atan2 c d
                 in if angle < 0
                       then angle + 2 * pi
                       else angle


sortByAngle :: [(Double, Double)] -> (Double, Double) -> [(Double, Double)]
sortByAngle [] _ = []
sortByAngle [a] _ = [a]
sortByAngle (a:as) z = a: sortBy f as
    where f xy1 xy2 = let angle1 = getAngle z a xy1
```

Line: 50 Col: 1

Upload Code as File Test against custom input Run Code Submit Code