Codility_

## Candidate Report: training2UXCX2-4VE

Check out Codility training tasks

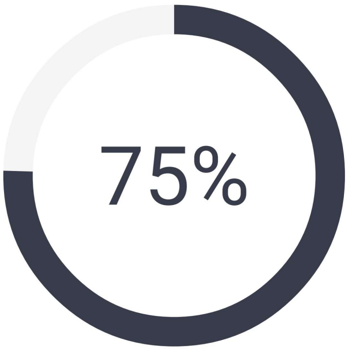
Test Name:

Summary Review (0) Timeline

### Tasks summary

| Task | Time spent | Score |
|---|---|---|
| **ChocolatesByNumbers**<br>Java 8 | 1 min | 75% |

### Total score



## Tasks Details

Easy

1. **ChocolatesByNumbers**
There are N chocolates in a circle. Count the number of chocolates you will eat.

| Task Score | Correctness | Performance |
|---|---|---|
| 75% | 100% | 50% |

### Task description

Two positive integers N and M are given. Integer N represents the number of chocolates arranged in a circle, numbered from 0 to N − 1.

You start to eat the chocolates. After eating a chocolate you leave only a wrapper.

You begin with eating chocolate number 0. Then you omit the next M − 1 chocolates or wrappers on the circle, and eat the following one.

More precisely, if you ate chocolate number X, then you will next eat the chocolate with number (X + M) modulo N (remainder of division).

You stop eating when you encounter an empty wrapper.

For example, given integers N = 10 and M = 4. You will eat the following chocolates: 0, 4, 8, 2, 6.

The goal is to count the number of chocolates that you will eat, following the above rules.

Write a function:

```
class Solution { public int solution(int N, int M); }
```

that, given two positive integers N and M, returns the number of chocolates that you will eat.

For example, given integers N = 10 and M = 4. the function should return 5, as explained above.

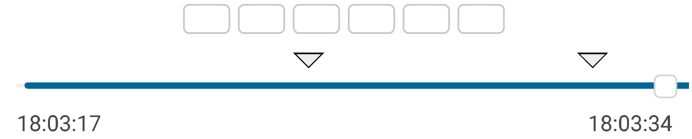
Write an **efficient** algorithm for the following assumptions:

- N and M are integers within the range [1..1,000,000,000].

### Solution

| | |
|---|---|
| Programming language used: | Java 8 |
| Total time used: | 1 minutes |
| Effective time used: | 1 minutes |
| Notes: | *not defined yet* |

### Task timeline



Code: 18:03:34 UTC, java, final, score: 75

show code in pop-up

```
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
    public int solution(int N, int M) {//%75 because of
```

```
                // write your code in Java SE 8
                int count = 1, mod = M%N;

                while(mod!=0){
                    count ++;
                    mod = (M+mod)%N;
                }
                return count;
            }
}
```

## Analysis summary

The following issues have been detected: timeout errors.

For example, for the input `(1000000000, 1)` the solution exceeded the time limit.

## Analysis ?

| expand all | Example tests | |
|---|---|---|
| ▶ | example<br>example test | ✓ OK |
| expand all | Correctness tests | |
| ▶ | extreme_small<br>very small N and M | ✓ OK |
| ▶ | simple<br>simple test, N = 24, M = 18 | ✓ OK |
| ▶ | small1<br>small tests | ✓ OK |
| ▶ | small2<br>small tests | ✓ OK |
| expand all | Performance tests | |
| ▶ | medium<br>medium tests | ✓ OK |
| ▶ | large<br>large tests | ✗ TIMEOUT ERROR<br>running time: 1.420 sec., time limit: 0.100 sec. |
| ▶ | large2<br>N = (3**9)*(2**14), M=(2**14)*(2**14) | ✓ OK |
| ▶ | extreme_large<br>maximal and minimal values | ✗ TIMEOUT ERROR<br>Killed. Hard limit reached: 6.000 sec. |

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.