codility

Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

100 out of 100 points

100%

Tasks in Test

	Time Spent	Task Score
FrogJmp Submitted in: Java 8	18 min	100%

TASKS DETAILS

EASY

1. FrogJmp
Count minimal number of jumps from position X to Y.

Task Score	Correctness	Performance
100%	100%	100%

Task description

A small frog wants to get to the other side of the road. The frog is currently located at position X and wants to get to a position greater than or equal to Y. The small frog always jumps a fixed distance, D.

Count the minimal number of jumps that the small frog must perform to reach its target.

Write a function:

```
class Solution { public int solution(int X, int Y, int D); }
```

that, given three integers X, Y and D, returns the minimal number of jumps from position X to a position equal to or greater than Y.

For example, given:

```
X = 10
Y = 85
D = 30
```

the function should return 3, because the frog will be positioned as follows:

- after the first jump, at position 10 + 30 = 40
- after the second jump, at position 10 + 30 + 30 = 70
- after the third jump, at position 10 + 30 + 30 + 30 = 100

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

- X, Y and D are integers within the range [1..1,000,000,000];
- X ≤ Y.

Solution

Programming language used:	Java 8
Total time used:	18 minutes
Effective time used:	18 minutes
Notes:	*not defined yet*

Task timeline

15:49:01 16:06:24

Code: 16:06:24 UTC, java, final, score: **100** show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8      public static int solution(int x,int y,int d ) {
9
```

```
10                  int s = y - x;
11                  int j0 = s/d;
12
13                  int t = s%d == 0 ? 0 : 1;
14
15                  return j0 + t;
16          }
17  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(1)**

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	simple1 simple test	✓ OK
▶	simple2	✓ OK
▶	extreme_position no jump needed	✓ OK
▶	small_extreme_jump one big jump	✓ OK
expand all	Performance tests	
▶	many_jump1 many jumps, D = 2	✓ OK
▶	many_jump2 many jumps, D = 99	✓ OK
▶	many_jump3 many jumps, D = 1283	✓ OK
▶	big_extreme_jump maximal number of jumps	✓ OK
▶	small_jumps many small jumps	✓ OK