

**Congratulations**

You have completed a Codility training test.

**Tweet this!**

I scored 100% in #java on @Codility!  
[https://codility.com/demo/take-sample-test/frog\\_jump/](https://codility.com/demo/take-sample-test/frog_jump/)

Sign up for our newsletter!

Like us on Facebook!

## Training ticket

**Session**

ID: trainingJW5QA4-EN6  
Time limit: 120 min.

**Status: closed**

Created on: 2016-06-19 17:34 UTC  
Started on: 2016-06-19 17:34 UTC  
Finished on: 2016-06-19 17:36 UTC

**Tasks in test**

1

**FrogJump**

Submitted in: Java

**Correctness**

100%

**Performance**

100%

**Task score**

100%

**Test score ?**

# 100%

100 out of 100 points

EASY

**1. FrogJump**

Count minimal number of jumps from position X to Y.

**score: 100 of 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  $10 + 30 + 30 + 30 = 100$

Assume that:

**Solution**

Programming language used: Java

Total time used: 2 minutes

?

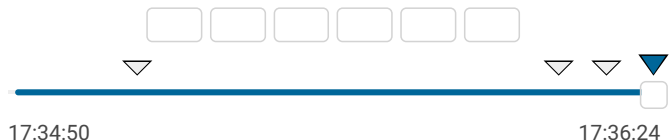
Effective time used: 2 minutes

?

Notes: *not defined yet*

**Task timeline**

?



Code: 17:36:24 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 {
```

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

Complexity:

- expected worst-case time complexity is  $O(1)$ ;
- expected worst-case space complexity is  $O(1)$ .

Copyright 2009–2016 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```
8 public int solution(int X, int Y, int D) {
9     int distance = Y - X;
10    double jumpsNeeded = (double) distance / D;
11    return (int) Math.ceil(jumpsNeeded);
12 }
13 }
```

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

Training center