Codility_

Candidate Report: training57KCJX-9A5

Test Name:

Summary   Timeline   Feedback

Tasks summary

Task	Time spent	Score
FrogJmp C#	3 min	100%

Total score

100%

Tasks Details

		Task Score	Correctness	Performance
Easy	1. **FrogJmp** Count minimal number of jumps from position X to Y.	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
```

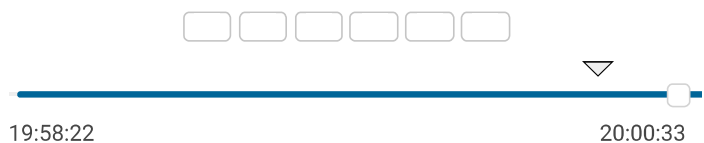the function should return 3, because the frog will be positioned as follows:

- 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

Solution

Programming language used:	C#
Total time used:	3 minutes
Effective time used:	3 minutes
Notes:	*not defined yet*

Task timeline

19:58:22   20:00:33

Code: 20:00:33 UTC, cs, final,   show code in pop-up
score: **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.

```
1   using System;
2   // you can also use other imports, for example:
3   // using System.Collections.Generic;
4
5   // you can write to stdout for debugging purposes, e.g.
6   // Console.WriteLine("this is a debug message");
7
8   class Solution {
9       public int solution(int X, int Y, int D)
10          {
11              if (X == Y)
12                  return 0;
13              var distance =(double)( Y - X );
14              double d = Math.Ceiling( distance / (double)D);
15              return System.Convert.ToInt32(d);
16          }
17  }
```

Analysis summary

The solution obtained perfect score.

Analysis ❓

Detected time complexity:

O(1)

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