**Jump To The Flag**

QUESTION DESCRIPTION

A climber is trying to reach a flag that is some height above the ground. In the attempt to reach the flag, the climber can make any number of jumps up the rock wall where the flag is mounted. Movements can only be made up the wall, and the climber must end at exactly the height of the flag.

There are 2 types of jumps:

1. A jump of height 1.

2. A jump of height bigJump.

Determine the minimum number of jumps it will take the climber to reach the flag's exact height.

**Example**

flagHeight = 8

bigJump = 3

The climber starts at height 0, takes two jumps of height bigJump and two of height 1 to reach exactly 8 units in 4 jumps.

**Function Description**

Complete the function jumps in the editor below.

jumps has the following parameter(s):

int flagHeight: an integer, the flag height

int bigJump: an integer, the height of the second type of jump

**Returns**:

int: an integer, the minimum number of jumps necessary

**Constraints**

* 1 ≤ bigJump, flagHeight ≤ 10

**Input Format for Custom Testing**

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer flagHeight.

The second line contains an integer bigJump.

**Sample Case 0**

**Sample Input**

STDIN Function

3 → flagHeight = 3

1 → bigJump = 1

**Sample Output**

3

**Explanation**

The climber can only jump 1 unit or bigJump units. With bigJump = 1, the climber can only make 1-unit jumps. It will take 3 jumps to reach the flag.

**Sample Case 1**

**Sample Input**

STDIN Function

3 → flagHeight = 3

2 → bigJump = 2

**Sample Output**

2

**Explanation**

The climber will jump bigJump = 2 units and then 1 unit to reach the flag with 2 jumps.

**Sample Case 2**

**Sample Input**

STDIN Function

3 → flagHeight = 3

3 → bigJump = 3

**Sample Output**

1

**Explanation**

The climber will make 1 jump bigJump = 3 units up the wall to reach the flag.

**Hint 1**

What are the most advantageous jumps to take? Answer: bigJump

**Hint 2**

How many of those jumps can you take? Answer: floor(flagHeight/bigJump)

**Solution**

Concepts covered: This problem tests the candidates on maths.

**Optimal Solution:**

The solution to this problem is the minimum value of x + y, such that x \* bigJump + y \* 1 = flagHeight

It is always optimal to take as many big jumps possible and take single jumps for the remainder. The solution to this problem is x = floor(flagHeight / bigJump) and y = flagHeight % bigJump.

def jumps(flagHeight, bigJump):

# Integer division returns the floor of the division

complete\_jump = flagHeight // bigJump

rem\_jump = flagHeight % bigJump

return complete\_jump + rem\_jump

**CANDIDATE ANSWER**

Language used: Java 8

class Result {

/\*

\* Complete the 'jumps' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER flagHeight

\* 2. INTEGER bigJump

\*/

public static int jumps(int flagHeight, int bigJump) {

}

}