*Bad programmers worry about the code. Good programmers worry about data structures and their relationships*

*- Linus Torvalds*

Dear readers, I hope that you are now comfortable in writing if-else statements, for and while loops, and the rest of the Java programming basics.

I welcome you to the new section ‘[*Function and Arrays*](https://www.pepcoding.com/resources/online-java-foundation/function-and-arrays/)’ of our online java foundation course. It will be a lot of fun to explore how should we modularize our code using functions, and use the most vital data structure - Arrays.

Do watch the [Introduction To Functions](https://www.pepcoding.com/resources/online-java-foundation/function-and-arrays/introduction-to-functions-official/video) video before moving on to this question.

***Problem Statement***

Given a number *n*, and a digit *d*, you are required to calculate the frequency of digit *d* in number *n*.

Important Links: [Problem Link](https://www.pepcoding.com/resources/online-java-foundation/function-and-arrays/digit-frequency-official/ojquestion), [Question Video](https://www.youtube.com/watch?v=TJvWrEMuO5k), [Solution Video](https://www.youtube.com/watch?v=U3VZNsMJpD4)

***Deducing Algorithm***

So, you know how to extract digits from a given number, right? Please remember how you had printed all digits of a number line-wise in this [problem](https://www.pepcoding.com/resources/online-java-foundation/getting-started/digits-of-a-number-official/ojquestion). I request you to try to apply the same logic in this problem yourself before reading further.

I hope that you have tried implementing it. So, since we need to count the number of times *d* occurs in *n,* we will make a *counter* variable. We will keep extracting digits from the number *n by repeated division by 10* (until it becomes 0)and check whether the current extracted digit is equal to d or not. If both are equal, then we will increment the *counter* by 1, otherwise, continue for the next iteration.

***Pseudo Code/ Algorithm***

* Set *counter* = 0
* Extract Digits of number *n* until *n* becomes 0
  + If the current digit extracted is the same as the digit d, update *counter* by 1.
* Return *counter* variable

Seems simple, right? Do give it a try before reading the code.

***Implementation (Java)***

import java.util.\*;

import java.lang.\*;

import java.io.\*;

public class Main {

public static void main(String[] args) {

Scanner scn = *new* Scanner(System.in);

int n = scn.nextInt();

int d = scn.nextInt();

int f = getDigitFrequency(n, d);

System.out.println(f);

}

public static int getDigitFrequency(int n, int d) {

int counter = 0;

*// initialize digit counter to 0*

*while*(n>0)

{

int temp = n%10; *// Extract Digits*

n = n/10; *// Divide N by 10*

*if*(temp == d) *// Logic Applied*

counter++;

}

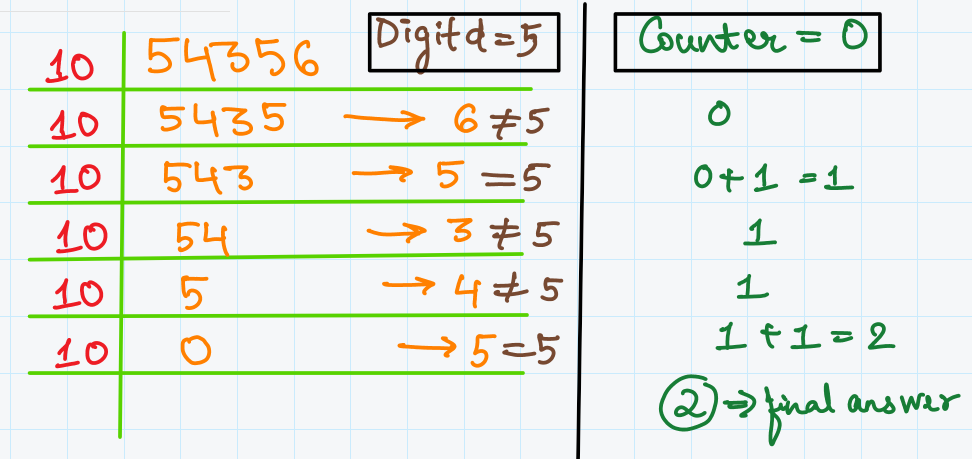
*return* counter;

}

}

This code is written and explained by Mr. Sumeet Malik in [this video](https://www.youtube.com/watch?v=U3VZNsMJpD4) from *[1:44, 3:30]*. Please refer to it if you are stuck somewhere.

***Sample Test Case***

****

Mr. Sumeet Malik has taken this test case to explain how the algorithm will work in the same video from *[1:07, 1:43]*.

***Time & Space Complexity Analysis***

Come on friend, I know you can answer this. Give it a try!

Time Complexity - We are extracting digits of number *n* which will take **O(*log10**n*)** time as there can be maximum *floor*(*log10**n*) digits in a number *n*.

Space Complexity - We just need to store the count of matched digits in an integer variable *counter,* hence **O(1)** auxiliary space is required.

I hope, you enjoyed solving the problem with me. Let’s meet now to deep dive into how the number system works from the next article.

*Article Written By: Archit Aggarwal*