Java Programs

1. Java's System.out.printf function can be used to print formatted output. The purpose of this exercise is to test your understanding of formatting output using printf.

To get you started, a portion of the solution is provided for you in the editor; you must format and print the input to complete the solution.

**Input Format**

Every line of input will contain a String followed by an integer.  
Each String will have a maximum of  10 alphabetic characters, and each integer will be in the inclusive range from 0 to 999 .

**Output Format**

In each line of output there should be two columns:  
The first column contains the String and is left justified using exactly  15 characters.  
The second column contains the integer, expressed in exactly  3 digits; if the original input has less than three digits, you must pad your output's leading digits with zeroes.

**Sample Input**

java 100

cpp 65

python 50

**Sample Output**

================================

java 100

cpp 065

python 050

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Java has 8 primitive data types; *char, boolean, byte, short, int, long, float, and double*. For this exercise, we'll work with the primitives used to hold integer values (*byte, short, int,* and *long*):

* A *byte* is an 8-bit signed integer.
* A *short* is a 16-bit signed integer.
* An *int* is a 32-bit signed integer.
* A *long* is a 64-bit signed integer.

Given an input integer, you must determine which primitive data types are capable of properly storing that input.

To get you started, a portion of the solution is provided for you in the editor.

**Reference:** <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

**Input Format**

The first line contains an integer,  **T** , denoting the number of test cases.  
Each test case,  **T** , is comprised of a single line with an integer,   **n**, which can be arbitrarily large or small.

**Output Format**

For each input variable  **n** and appropriate primitive  **data type**, you must determine if the given primitives are capable of storing it. If yes, then print:

n can be fitted in:

\* dataType

If there is more than one appropriate data type, print each one on its own line and order them by size (i.e.: ).

**Byte< short <int< long**

If the number cannot be stored in one of the four aforementioned primitives, print the line:

n can't be fitted anywhere.

**Sample Input**

5

-150

150000

1500000000

213333333333333333333333333333333333

-100000000000000

**Sample Output**

-150 can be fitted in:

\* short

\* int

\* long

150000 can be fitted in:

\* int

\* long

1500000000 can be fitted in:

\* int

\* long

213333333333333333333333333333333333 can't be fitted anywhere.

-100000000000000 can be fitted in:

\* long

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. The challenge here is to read  **n** lines of input until you reach EOF, then number and print all   **n** lines of content.

**Hint:** Java's Scanner.hasNext() method is helpful for this problem.

**Input Format**

Read some unknown  **n**  lines of input from stdin(System.in) until you reach EOF; each line of input contains a non-empty String.

**Output Format**

For each line, print the line number, followed by a single space, and then the line content received as input.

**Sample Input**

Hello world

I am a file

Read me until end-of-file.

**Sample Output**

1 Hello world

2 I am a file

3 Read me until end-of-file.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\*\*\*\*\*ALL THE BEST \*\*\*\*\*\*\*\*\*\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_