**Java Datatypes**

<https://www.hackerrank.com/challenges/java-datatypes/problem>

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 *dataType*, 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

**Explanation**

*-150* can be stored in a *short*, an *int*, or a *long*.

*213333333333333333333333333333333333* is very large and is outside of the allowable range of values for the primitive data types discussed in this problem.