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

Output Format

For each input variable and appropriate primitive , 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.:).

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 anywhe
-100000000000000 can be fitted in:
* long
```

Explanation

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

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

Change Theme Language Java 8

```
1  import java.util.*;
2
3  class Solution{
4      public static void main(String []argh)
5      {
6          Scanner sc = new Scanner(System.in);
7          int t=sc.nextInt();
8          for(int i=0;i<t;i++)
9          {
10
11             try
12             {
13                 long x=sc.nextLong();
14                 System.out.println(x+" can be fitted in:");
15                 if(x>=-128 && x<=127) {
16                     System.out.println("* byte");
17                 }
18                 if(x>=-32768 && x<=32767) {
19                     System.out.println("* short");
20                 }
21                 if(x>=-2147483648 && x<=2147483647) {
22                     System.out.println("* int");
23                 }
24                 if(x>-2147483648 || x<2147483647) {
25                     System.out.println("* long");
26                 }
27             }
28             catch(Exception e)
29             {
30                 System.out.println(sc.next()+" can't be fitted anywh
31             }
32
33         }
34         sc.close();
35     }
36 }
```

Line: 34 Col: 20

Upload Code as File Test against custom input Run Code Submit Code

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0
Test case 1
Test case 2
Test case 3

Compiler Message

Success

Input (stdin) Download

```
5
-150
150000
1500000000
213333333333333333333333333333333333
-100000000000000
```

Expected Output Download

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 anywhe
-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.

Change Theme Language Java 8

```
1  import java.util.*;
2
3  class Solution{
4      public static void main(String []argh)
5      {
6          Scanner sc = new Scanner(System.in);
7          int t=sc.nextInt();
8          for(int i=0;i<t;i++)
9          {
10
11             try
12             {
13                 long x=sc.nextLong();
14                 System.out.println(x+" can be fitted in:");
15                 if(x>=-128 && x<=127) {
16                     System.out.println("* byte");
17                 }
18                 if(x>=-32768 && x<=32767) {
19                     System.out.println("* short");
20                 }
21                 if(x>=-2147483648 && x<=2147483647) {
22                     System.out.println("* int");
23                 }
24                 if(x>-2147483648 || x<2147483647) {
25                     System.out.println("* long");
26                 }
27             }
28             catch(Exception e)
29             {
30                 System.out.println(sc.next()+" can't be fitted anywh
31             }
32
33         }
34         sc.close();
35     }
36 }
```

Line: 34 Col: 20

Test against custom input

Congratulations

You solved this challenge. Would you like to challenge your friends? Next Challenge

Compiler Message

Success

Input (stdin)

```
5
-150
150000
1500000000
213333333333333333333333333333333333
-100000000000000
```

Expected Output