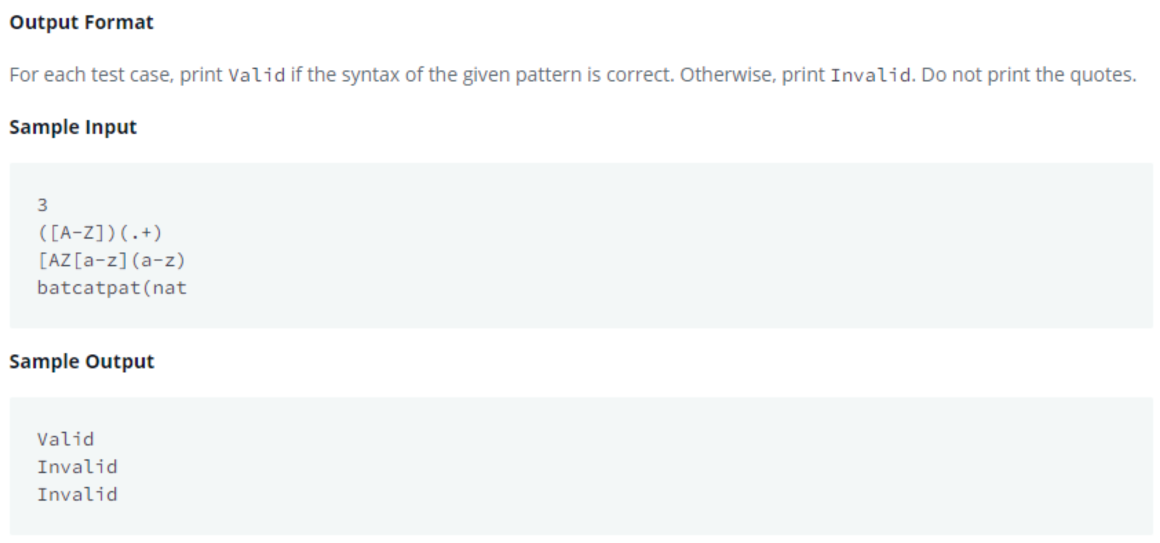
NAMA : JAMALUDIN

TYPE : EASY

TUGAS HACKERRANKS DAY 3

---

1. Question “Pattern Syntax Checker”



Output Format

For each test case, print `Valid` if the syntax of the given pattern is correct. Otherwise, print `Invalid`. Do not print the quotes.

Sample Input

```
3
([A-Z])(.+)
[AZ[a-z](a-z)
batcatpat(nat
```

Sample Output

```
Valid
Invalid
Invalid
```

Answer

```java
import java.util.Scanner;
import java.util.regex.*;

public class Solution
{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int testCases = Integer.parseInt(in.nextLine());
        while(testCases>0){
            String pattern = in.nextLine();
            try {
                Pattern.compile(pattern);
                System.out.println("Valid");
            } catch (PatternSyntaxException e) {
                System.out.println("Invalid");
            }
            testCases--;
        }
    }
}
```

2. <u>“Java Factory”</u>

According to Wikipedia, a factory is simply an object that returns another object from some other method call, which is assumed to be "new".

In this problem, you are given an interface Food. There are two classes Pizza and Cake which implement the Food interface, and they both contain a method getType().

The main function in the Main class creates an instance of the FoodFactory class. The FoodFactory class contains a method getFood(String) that returns a new instance of Pizza or Cake according to its parameter.

You are given the partially completed code in the editor. Please complete the FoodFactory class.

Sample Input 1

```
cake
```

Sample Output 1

```
The factory returned class Cake
Someone ordered a Dessert!
```

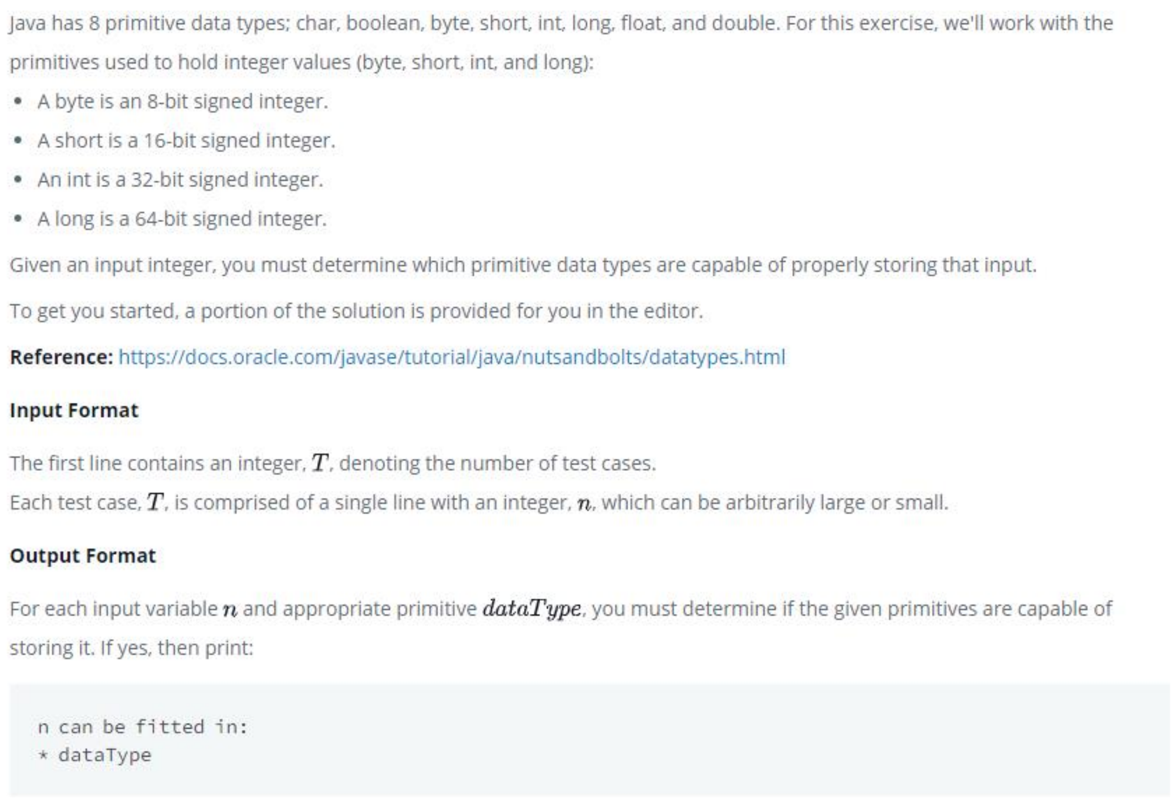
Sample Input 2

```
pizza
```

```
import java.util.*;
import java.security.*;
interface Food {
     public String getType();
    }
    class Pizza implements Food {
     public String getType() {
           return "Someone ordered a Fast Food!";
           }
          }

          class Cake implements Food {

           public String getType() {
           return "Someone ordered a Dessert!";
           }
          }
          class FoodFactory {
              public Food getFood(String order) {

```

```
    switch (order){
    case "pizza": return new Pizza();
         case "cake" : return new Cake();
         default : return null;
    }
    }//End of getFood method
```

3. Question “Java DataTypes”


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
```

Answer

```
import java.util.*;
import java.io.*;



class Solution{
    public static void main(String []argh)
    {



        Scanner sc = new Scanner(System.in);
        int t=sc.nextInt();

        for(int i=0;i<t;i++)
        {

            try
            {
                long x=sc.nextLong();
                System.out.println(x+" can be fitted in:");
                if(x>=-128 && x<=127)System.out.println("* byte");

                if(x>=-(Math.pow(2,15)) && x<=(Math.pow(2,15)-1))
                System.out.println("* short");

                if(x>=-(Math.pow(2,31)) && x<=(Math.pow(2,31)-1))
                System.out.println("* int");

                if(x>=-(Math.pow(2,63)) && x<=(Math.pow(2,63)-1))
                System.out.println("* long");
            }
            catch(Exception e)
            {
                System.out.println(sc.next()+" can't be fitted anywhere
.");
            }

        }
    }
}
```

4. Question "Java Static Initializer Block"

Static initialization blocks are executed when the class is loaded, and you can initialize static variables in those blocks.

It's time to test your knowledge of Static initialization blocks. You can read about it here.

You are given a class Solution with a main method. Complete the given code so that it outputs the area of a parallelogram with breadth B and height H. You should read the variables from the standard input.

If $B \leq 0$ or $H \leq 0$, the output should be "java.lang.Exception: Breadth and height must be positive" without quotes.

Input Format

There are two lines of input. The first line contains B: the breadth of the parallelogram. The next line contains H: the height of the parallelogram.

Constraints

- $-100 \leq B \leq 100$
- $-100 \leq H \leq 100$

Output Format

If both values are greater than zero, then the main method must output the area of the parallelogram. Otherwise, print "java.lang.Exception: Breadth and height must be positive" without quotes.

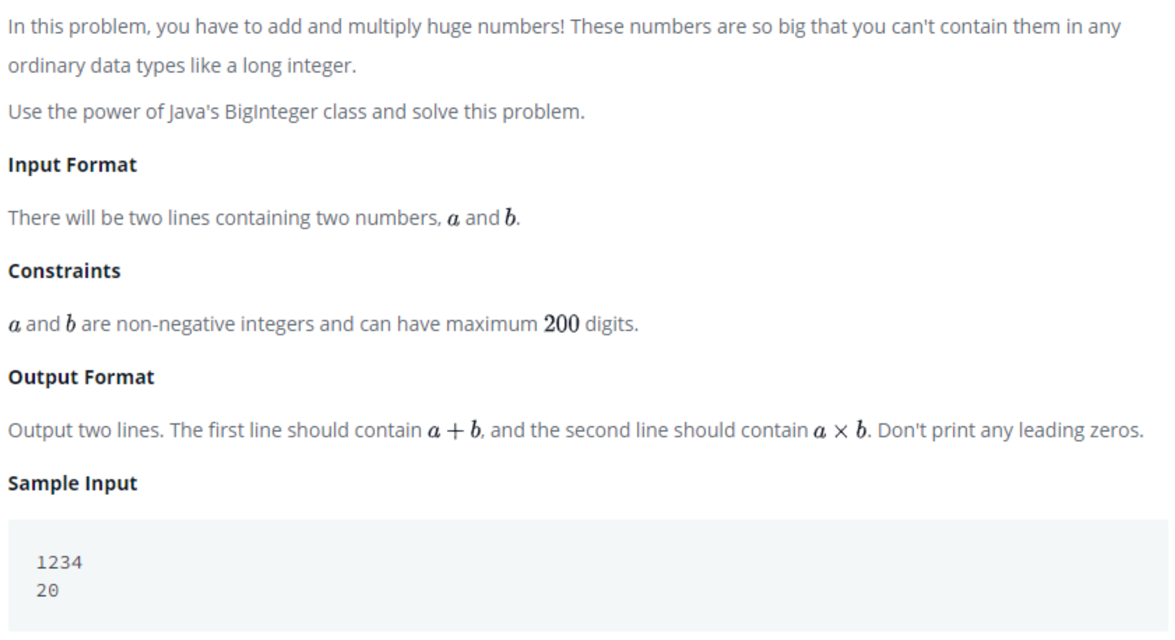
Sample input 1

```
1
3
```

Answer

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
static boolean flag = true; static int B,H;

static{

Scanner scan = new Scanner(System.in); B = scan.nextInt(); scan.nextLine(); H = scan.nextInt(); scan.close(); if(B>0 && H>0){ flag = true;

} else if((B<=0 && H>=0)||(B>=0 && H<=0)){ flag=false; System.out.println("java.lang.Exception: Breadth and height must be positive"); } else { flag=false; System.out.println("java.lang.Exception: Breadth and height must be positive"); }

}
public static void main(String[] args){
        if(flag){
            int area=B*H;
            System.out.print(area);
        }

    }//end of main

}//end of class
```

5. Question “Java BigInteger”



In this problem, you have to add and multiply huge numbers! These numbers are so big that you can't contain them in any ordinary data types like a long integer.

Use the power of Java's BigInteger class and solve this problem.

Input Format

There will be two lines containing two numbers, a and b.

Constraints

a and b are non-negative integers and can have maximum 200 digits.

Output Format

Output two lines. The first line should contain $a + b$, and the second line should contain $a \times b$. Don't print any leading zeros.

Sample Input

```
1234
20
```

<u>Answer</u>

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         BigInteger bi1 = new BigInteger(sc.next());
         BigInteger bi2 = new BigInteger(sc.next());
```

```
        BigInteger  bi3, bi4;
        bi3 = bi1.add(bi2);
        bi4 = bi1.multiply(bi2);
        System.out.println( bi3);
        System.out.println( bi4);


}
}
```

6. Question “Java Generic”

Generic methods are a very efficient way to handle multiple datatypes using a single method. This problem will test your knowledge on Java Generic methods.

Let's say you have an integer array and a string array. You have to write a **single** method printArray that can print all the elements of both arrays. The method should be able to accept both integer arrays or string arrays.

You are given code in the editor. Complete the code so that it prints the following lines:

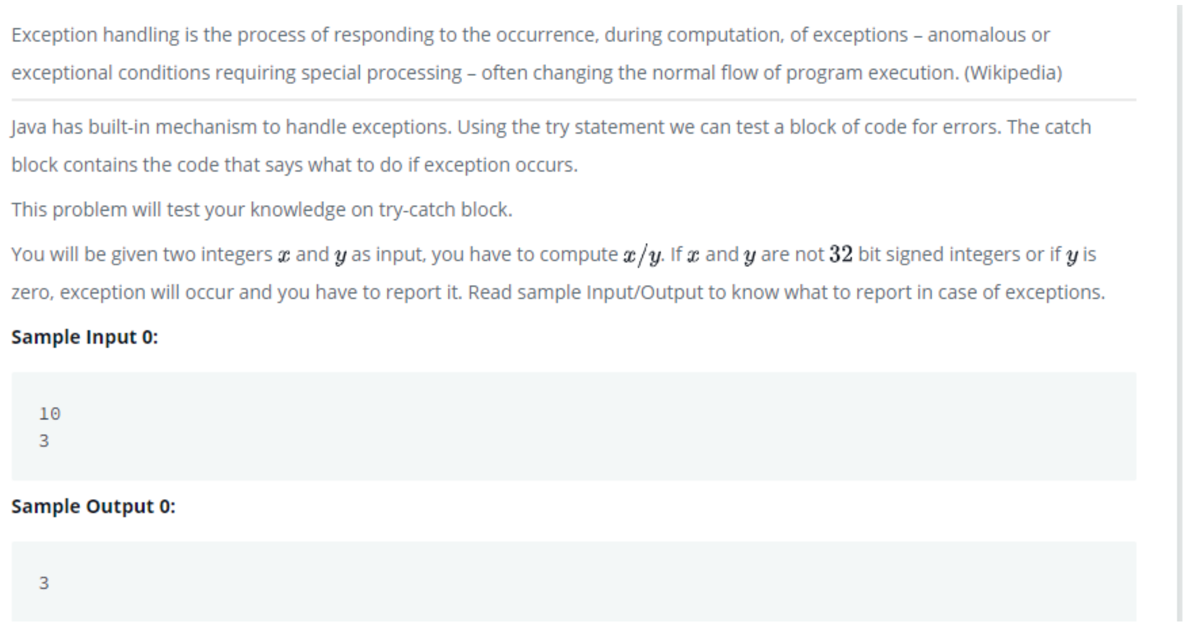
```
1
2
3
Hello
World
```

Do not use method overloading because your answer will not be accepted.

Answer

```
import java.io.IOException;
import java.lang.reflect.Method;

class Printer
{
   public <T> void printArray(T[] array){
    for(T item: array){
        System.out.println(item);
    }
}

}

public class Solution {
    public static void main( String args[] ) {
        Printer myPrinter = new Printer();
        Integer[] intArray = { 1, 2, 3 };
        String[] stringArray = {"Hello", "World"};
        myPrinter.printArray(intArray);
        myPrinter.printArray(stringArray);
        int count = 0;

        for (Method method : Printer.class.getDeclaredMethods()) {
            String name = method.getName();

            if(name.equals("printArray"))
                count++;
        }

        if(count > 1)System.out.println("Method overloading is not allo
wed!");

    }
}
```

7. Question “Java Exception Handling (Ttry-Catch)”



Exception handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. (Wikipedia)

Java has built-in mechanism to handle exceptions. Using the try statement we can test a block of code for errors. The catch block contains the code that says what to do if exception occurs.

This problem will test your knowledge on try-catch block.

You will be given two integers x and y as input, you have to compute x/y. If x and y are not 32 bit signed integers or if y is zero, exception will occur and you have to report it. Read sample Input/Output to know what to report in case of exceptions.

Sample Input 0:

```
10
3
```

Sample Output 0:

```
3
```

Answer

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
```

```java
public class Solution {

    public static void main(String[] args) {
    try{
            Scanner sc = new Scanner(System.in);
            int x = sc.nextInt();
            int y = sc.nextInt();
            if(y==0)
                throw  new ArithmeticException("/ by zero");
            else
                System.out.println(x/y);
        }
        catch(InputMismatchException e){
            System.out.println(e.getClass().getName());
        }

        catch(ArithmeticException e){
            System.out.println(e);
        }}
}
```

NAMA : JAMALUDIN

TYPE : MEDIUM

---

1. Question “Java BigDecimal”

Java's BigDecimal class can handle arbitrary-precision signed decimal numbers. Let's test your knowledge of them!

Given an array, s, of n real number strings, sort them in descending order — but wait, there's more! Each number must be printed in the exact same format as it was read from stdin, meaning that $.1$ is printed as $.1$, and 0.1 is printed as 0.1. If two numbers represent numerically equivalent values (e.g., $.1 \equiv 0.1$), then they must be listed in the same order as they were received as input).

Complete the code in the unlocked section of the editor below. You must rearrange array s's elements according to the instructions above.

Input Format

The first line consists of a single integer, n, denoting the number of integer strings.
Each line i of the n subsequent lines contains a real number denoting the value of s_i.

Constraints

- $1 \leq n \leq 200$
- Each s_i has at most 300 digits.

Output Format

Locked stub code in the editor will print the contents of array s to stdout. You are only responsible for reordering the array's elements.

Answer

```java
import java.math.BigDecimal;
import java.util.*;
class Solution{

    public static void main(String []args){
        //Input
        Scanner sc= new Scanner(System.in);
        int n=sc.nextInt();
        String []s=new String[n+2];
        for(int i=0;i<n;i++){
            s[i]=sc.next();
        }
        sc.close();

       Arrays.sort(s, new Comparator<String>() {
        @Override
        public int compare(String o1, String o2) {
        if (o1 == null || o2 == null) {
            return 0;
        }
        BigDecimal o1bd = new BigDecimal(o1);
        BigDecimal o2bd = new BigDecimal(o2);
        return o2bd.compareTo(o1bd);
        }
    });
        //Output
        for(int i=0;i<n;i++)
        {
            System.out.println(s[i]);
        }
    }

}
```