JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

JGI Knowledge Campus, Jayanagar 9th Block, Bengaluru – 69

Department of Computer Science and IT

MCA PROGRAMME

ACTIVITY – 2

Object Oriented Programming using Java

23MCAC202

PROBLEM SOLVING IN PROGRAMMING PLATFORM

Submitted By:
Rahul Rao M
2nd SEM
MCA – ISMS
USN:24MCAR0210

Submitted to:
Dr. Kamalraj R
Professor
Dept. of CS & IT

Questions Solved on Hacker Rank and LeetCode:

Java Basics

1. Welcome to Java
2. Java Stdin and Stdout I
3. Java If-Else
4. Java Stdin and Stdout II
5. Java Output Formatting
6. Java Loop I
7. Java Loop II
8. Java Data Type
9. Java End of File
10. Java Static Initializer Block
11. Java Int to String
12. Java Date and Time
13. Java Currency Formatter

Java Strings & Regex

14. Java Strings Introduction
15. Java Substring
16. Java Substring Comparisons
17. Java String Reverse
18. Java Anagrams
19. Java String Tokens
20. Pattern Syntax Checker
21. Java Regex
22. Java Regex 2 - Duplicate Words
23. Valid Username Regular Expression
24. Tag Content Extractor

Java Big Numbers & Primality

25. Java BigDecimal
26. Java Primality Test
27. Java BigInteger

Java Arrays & Collections

28. Java 1D Array
29. Java 2D Array
30. Java Subarray
31. Java ArrayList
32. Java 1D Array (Part 2)
33. Java List
34. Java Map
35. Java Stack
28. Java Generics
29. Java Comparator
30. Java Sort
31. Java Dequeue
32. Java BitSet

Java OOP & Advanced Topics

41. Java Inheritance I
42. Java Inheritance II
43. Java Abstract Class
44. Java Interface
45. Java Method Overriding
46. Java Method Overriding 2 (Super Keyword)
47. Java Instanceof Keyword
48. Java Iterator

Java Exception Handling & Reflection

49. Java Exception Handling (Try-catch)
50. Java Exception Handling
51. Java Varargs - Simple Addition
52. Java Reflection - Attributes
53. Can You Access?
54. Prime Checker

Design Patterns & Functional Programming

55. Java Factory Pattern

Rahul Rao M

@rahulroxx2002

Personal Information

rahulroxx2002@gmail.com

+91-7619400520

India

My Resume

Rahul Rao M- Resume.pdf

EEO settings

Complete your profile

Add your missing details

20%

This data will be helpful to auto-fill your job applications

My Badges

My Certifications

Java

62/64 challenges solved

Rank: 15258 | Points: 911

Welcome to Java!

Easy, Max Score: 3, Success Rate: 97.06%

Solved

Java Stdin and Stdout I

Easy, Java (Basic), Max Score: 5, Success Rate: 96.80%

Solved

Java If-Else

Easy, Java (Basic), Max Score: 10, Success Rate: 91.58%

Solved

Java Stdin and Stdout II

Easy, Java (Basic), Max Score: 10, Success Rate: 93.14%

Solved

Java Output Formatting

Easy, Java (Basic), Max Score: 10, Success Rate: 96.61%

Solved

Java Loops I

Easy, Java (Basic), Max Score: 10, Success Rate: 97.69%

Solved

Java Loops II

Easy, Java (Basic), Max Score: 10, Success Rate: 97.34%

Solved

Java Datatypes

Easy, Java (Basic), Max Score: 10, Success Rate: 93.78%

Solved

Java End-of-file

Easy, Java (Basic), Max Score: 10, Success Rate: 97.93%

Solved

Java Static Initializer Block

Easy, Java (Basic), Max Score: 10, Success Rate: 96.10%

Solved

Java Int to String

Easy, Java (Basic), Max Score: 10, Success Rate: 97.86%

Solved

Java Date and Time

Easy, Java (Basic), Max Score: 15, Success Rate: 92.66%

Solved

Java Currency Formatter

Easy, Java (Basic), Max Score: 15, Success Rate: 96.29%

Solved

Java Strings Introduction

Easy, Java (Basic), Max Score: 5, Success Rate: 93.92%

Solved

Java Substring

Easy, Java (Basic), Max Score: 5, Success Rate: 98.93%

Solved

Java Substring Comparisons

Easy, Java (Basic), Max Score: 10, Success Rate: 92.08%

Solved

Java String Reverse

Easy, Java (Basic), Max Score: 10, Success Rate: 97.88%

Solved

Java Anagrams

Easy, Java (Basic), Max Score: 10, Success Rate: 93.38%

Solved

Java String Tokens

Easy, Java (Basic), Max Score: 15, Success Rate: 82.91%

Solved

Pattern Syntax Checker

Easy, Java (Basic), Max Score: 20, Success Rate: 97.52%

Solved

Java Regex

Medium, Java (Intermediate), Max Score: 25, Success Rate: 92.43%

Solved

Java Regex 2 - Duplicate Words

Medium, Java (Basic), Max Score: 25, Success Rate: 90.11%

Solved

Valid Username Regular Expression

Easy, Max Score: 20, Success Rate: 95.59%

Solved

Tag Content Extractor

Medium, Java (Basic), Max Score: 20, Success Rate: 95.25%

Solved

Java BigDecimal

Medium, Java (Basic), Max Score: 20, Success Rate: 94.58%

Solved

Java Primality Test

Easy, Java (Basic), Max Score: 20, Success Rate: 91.14%

Solved

Java BigInteger

Easy, Java (Basic), Max Score: 10, Success Rate: 97.67%

Solved

Java 1D Array

Easy, Java (Basic), Max Score: 5, Success Rate: 97.22%

Solved

Java 2D Array

Easy, Java (Basic), Max Score: 10, Success Rate: 93.07%

Solved

Java Subarray

Easy, Java (Basic), Max Score: 10, Success Rate: 97.03%

Solved

Java Arraylist

Easy, Java (Basic), Max Score: 10, Success Rate: 97.75%

Solved

Java 1D Array (Part 2)

Medium, Java (Basic), Max Score: 25, Success Rate: 73.49%

Solved

Java List

Easy, Java (Basic), Max Score: 15, Success Rate: 95.73%

Solved

Java Map

Easy, Java (Basic), Max Score: 10, Success Rate: 96.44%

Solved

Java Stack

Medium, Java (Basic), Max Score: 20, Success Rate: 92.47%

Solved

Java Generics

Easy, Java (Basic), Max Score: 15, Success Rate: 98.76%

Solved

Java Comparator

Medium, Java (Basic), Max Score: 10, Success Rate: 97.43%

Solved

Java Sort

Easy, Java (Basic), Max Score: 10, Success Rate: 94.82%

Solved

Java Dequeue

Medium, Problem Solving (Intermediate), Max Score: 20, Success Rate: 84.17%

Solved

Java BitSet

Easy, Java (Basic), Max Score: 20, Success Rate: 97.20%

Solved

Java Inheritance I

Easy, Java (Basic), Max Score: 5, Success Rate: 98.20%

Solved

Java Inheritance II

Easy, Java (Basic), Max Score: 10, Success Rate: 97.69%

Solved

Java Abstract Class

Easy, Java (Basic), Max Score: 10, Success Rate: 98.56%

Solved

Java Interface

Easy, Java (Basic), Max Score: 10, Success Rate: 98.15%

Solved

Java Method Overriding

Easy, Java (Basic), Max Score: 10, Success Rate: 98.81%

Solved

Java Method Overriding 2 (Super Keyword)

Easy, Java (Basic), Max Score: 10, Success Rate: 99.39%

Solved

Java Instanceof keyword

Easy, Java (Basic), Max Score: 10, Success Rate: 97.43%

Solved

Java Iterator

Easy, Java (Basic), Max Score: 15, Success Rate: 97.96%

Solved

Java Exception Handling (Try-catch)

Easy, Java (Basic), Max Score: 10, Success Rate: 94.87%

Solved

Java Exception Handling

Easy, Java (Basic), Max Score: 15, Success Rate: 95.47%

Solved

Java Varargs - Simple Addition

Easy, Java (Basic), Max Score: 15, Success Rate: 97.62%

Solved

Java Reflection - Attributes

Easy, Java (Basic), Max Score: 15, Success Rate: 89.00%

Solved

Can You Access?

Medium, Max Score: 15, Success Rate: 96.50%

Solved

Prime Checker

Medium, Java (Basic), Max Score: 25, Success Rate: 92.55%

Solved

Java Factory Pattern

Easy, Java (Basic), Max Score: 15, Success Rate: 97.91%

Solved

Java Singleton Pattern

Easy, Java (Basic), Max Score: 15, Success Rate: 97.23%

Solved

Java Visitor Pattern

Medium, Max Score: 40, Success Rate: 87.37%

Solved

Java Annotations

Medium, Max Score: 25, Success Rate: 94.15%

Solved

Covariant Return Types

Easy, Max Score: 20, Success Rate: 97.37%

Solved

Java Lambda Expressions

Medium, Max Score: 30, Success Rate: 90.14%

Solved

Java MD5

Medium, Max Score: 30, Success Rate: 98.18%

Solved

Java SHA-256

Medium, Max Score: 30, Success Rate: 96.45%

Solved

All My Submissions

Question	Status	Runtime	Language
Longest Common Prefix (/problems/longest-common-prefix)	**Accepted** (/submissions/detail/1617749156/)	0 ms	java
Roman to Integer (/problems/roman-to-integer)	**Accepted** (/submissions/detail/1617748900/)	5 ms	java
Integer to Roman (/problems/integer-to-roman)	**Accepted** (/submissions/detail/1617748607/)	3 ms	java
Regular Expression Matching (/problems/regular-expression-matching)	**Accepted** (/submissions/detail/1617748383/)	2 ms	java
Zigzag Conversion (/problems/zigzag-conversion)	**Accepted** (/submissions/detail/1617747870/)	4 ms	java
Longest Palindromic Substring (/problems/longest-palindromic-substring)	**Accepted** (/submissions/detail/1617747483/)	15 ms	java
Longest Substring Without Repeating Characters (/problems/longest-substring-without-repeating-characters)	**Accepted** (/submissions/detail/1617744966/)	6 ms	java
Count of Interesting Subarrays (/problems/count-of-interesting-subarrays)	**Accepted** (/submissions/detail/1617744642/)	37 ms	java
Count of Interesting Subarrays (/problems/count-of-interesting-subarrays)	**Wrong Answer** (/submissions/detail/1617743981/)	N/A	java
Count of Interesting Subarrays (/problems/count-of-interesting-subarrays)	**Wrong Answer** (/submissions/detail/1617700735/)	N/A	java
Add Two Numbers (/problems/add-two-numbers)	**Accepted** (/submissions/detail/1617700254/)	1 ms	java
Add Two Numbers (/problems/add-two-numbers)	**Compile Error** (/submissions/detail/1617699522/)	N/A	java
Add Two Numbers (/problems/add-two-numbers)	**Compile Error** (/submissions/detail/1617698834/)	N/A	java
Add Two Numbers (/problems/add-two-numbers)	**Compile Error** (/submissions/detail/1617698519/)	N/A	java
Two Sum (/problems/two-sum)	**Accepted** (/submissions/detail/1617695365/)	2 ms	java
Add Two Numbers II (/problems/add-two-numbers-ii)	**Accepted** (/submissions/detail/1198594336/)	16 ms	cpp
Trim a Binary Search Tree (/problems/trim-a-binary-search-tree)	**Accepted** (/submissions/detail/1198593721/)	9 ms	cpp

Newer Older

Q1. Welcome to Java

Welcome to the world of Java! In this challenge, we practice printing to stdout.

The code stubs in your editor declare a Solution class and a main method. Complete the main method by copying the two lines of code below and pasting them inside the body of your main method.

```
System.out.println("Hello, World.");
System.out.println("Hello, Java.");
```

Input Format

There is no input for this challenge.

Output Format

You must print two lines of output:

1. Print `Hello, World.` on the first line.
2. Print `Hello, Java.` on the second line.

Sample Output

```
Hello, World.
Hello, Java.
```

Submitted Code

Language: Java 7 — Open in editor

```
1 public class Solution {
2
3     public static void main(String[] args) {
4         /* Enter your code here. Print output to STDOUT. Your class should be named Solution. */
5         System.out.println("Hello, World.");
6         System.out.println("Hello, Java.");
7     }
8 }
9
```

Test case 0

Compiler Message

```
Success
```

Expected Output — Download

```
1 Hello, World.
2 Hello, Java.
```

Q2.Java Stdin and Stdout I

Most HackerRank challenges require you to read input from stdin (standard input) and write output to stdout (standard output).

One popular way to read input from stdin is by using the Scanner class and specifying the Input Stream as System.in. For example:

```
Scanner scanner = new Scanner(System.in);
String myString = scanner.next();
int myInt = scanner.nextInt();
scanner.close();

System.out.println("myString is: " + myString);
System.out.println("myInt is: " + myInt);
```

The code above creates a Scanner object named *scanner* and uses it to read a String and an int. It then closes the Scanner object because there is no more input to read, and prints to stdout using System.out.println(String). So, if our input is:

```
Hi 5
```

Our code will print:

```
myString is: Hi
myInt is: 5
```

Alternatively, you can use the BufferedReader class.

Task

In this challenge, you must read 3 integers from stdin and then print them to stdout. Each integer must be printed on a new line. To make the problem a little easier, a portion of the code is provided for you in the editor below.

Input Format

There are 3 lines of input, and each line contains a single integer.

Sample Input

```
42
100
125
```

Sample Output

```
42
100
125
```

Submitted Code

Language: Java 7 — Open in editor

```
        // Read three integers
        int a = scan.nextInt();
        int b = scan.nextInt();
        int c = scan.nextInt();

        scan.close();

        // Print the integers
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
42
100
125
```

Expected Output — Download

```
42
100
125
```

Q3.Java If-else

Task

Given an integer, n, perform the following conditional actions:

- If n is odd, print `Weird`
- If n is even and in the inclusive range of 2 to 5, print `Not Weird`
- If n is even and in the inclusive range of 6 to 20, print `Weird`
- If n is even and greater than 20, print `Not Weird`

Complete the stub code provided in your editor to print whether or not n is weird.

Input Format

A single line containing a positive integer, n.

Constraints

- $1 \leq n \leq 100$

Output Format

Print `Weird` if the number is weird; otherwise, print `Not Weird`.

Sample Input 0

```
3
```

Sample Output 0

```
Weird
```

Submitted Code

Language: Java 7 — Open in editor

```
        // Applying the conditions
        if (N % 2 != 0) {
            System.out.println("Weird");
        } else {
            if (N >= 2 && N <= 5) {
                System.out.println("Not Weird");
            } else if (N >= 6 && N <= 20) {
                System.out.println("Weird");
            } else if (N > 20) {
                System.out.println("Not Weird");
            }
        }
    }
}
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

Input (stdin) — Download

```
3
```

Expected Output — Download

```
Weird
```

Q4.Java Stdin and Stdout II

In this challenge, you must read an integer, a double, and a String from stdin, then print the values according to the instructions in the Output Format section below. To make the problem a little easier, a portion of the code is provided for you in the editor.

Note: We recommend completing Java Stdin and Stdout I before attempting this challenge.

Input Format

There are three lines of input:

1. The first line contains an integer.
2. The second line contains a double.
3. The third line contains a String.

Output Format

There are three lines of output:

1. On the first line, print `String:` followed by the unaltered String read from stdin.
2. On the second line, print `Double:` followed by the unaltered double read from stdin.
3. On the third line, print `Int:` followed by the unaltered integer read from stdin.

To make the problem easier, a portion of the code is already provided in the editor.

Note: If you use the nextLine() method immediately following the nextInt() method, recall that nextInt() reads integer tokens; because of this, the last newline character for that line of integer input is still queued in the input buffer and the next nextLine() will be reading the remainder of the integer line (which is empty).

Sample Input

```
42
3.1415
Welcome to HackerRank's Java tutorials!
```

Sample Output

```
String: Welcome to HackerRank's Java tutorials!
Double: 3.1415
Int: 42
```

Submitted Code

Language: Java 7 — Open in editor

```
        // Read integer
        int i = scan.nextInt();

        // Read double
        double d = scan.nextDouble();
        scan.nextLine(); // Consume the leftover newline

        // Read string
        String s = scan.nextLine();

        scan.close();

        // Print output in required format
        System.out.println("String: " + s);
        System.out.println("Double: " + d);
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Input (stdin) — Download

```
42
3.1415
Welcome to HackerRank's Java tutorials!
```

Expected Output — Download

```
String: Welcome to HackerRank's Java tutorials!
Double: 3.1415
Int: 42
```

Q5. Java output formatting

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 pac your output's leading digits with zeroes.

Sample Input

```
java 100
cpp 65
python 50
```

Sample Output

```
================================
java           100
cpp            065
python         050
================================
```

Explanation

Each String is left-justified with trailing whitespace through the first 15 characters. The leading digit of the integer is the 16^{th} character, and each integer that was less than 3 digits now has leading zeroes.

Submitted Code

Language: Java 7

Open in editor

```
public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("================================");
        for(int i=0;i<3;i++){
            String s1=sc.next();
            int x=sc.nextInt();
            //Complete this line
        System.out.printf("%-15s%03d%n", s1, x);
        }
        System.out.println("================================");
        sc.close();

    }
}
```

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

Success

Input (stdin) Download

```
java 100
cpp 65
python 50
```

Expected Output Download

```
================================
java           100
cpp            065
python         050
```

Q6. Java Loop I

Objective

In this challenge, we're going to use loops to help us do some simple math.

Task

Given an integer, N, print its first 10 multiples. Each multiple $N \times i$ (where $1 \leq i \leq 10$) should be printed on a new line in the form: `N x i = result`.

Input Format

A single integer, N.

Constraints

- $2 \leq N \leq 20$

Output Format

Print 10 lines of output; each line i (where $1 \leq i \leq 10$) contains the $result$ of $N \times i$ in the form: `N x i = result`.

Sample Input

```
2
```

Sample Output

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Submitted Code

Language: Java 7 — Open in editor

```
10
11 public class Solution {
12     public static void main(String[] args) throws IOException {
13         BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
14
15         int N = Integer.parseInt(bufferedReader.readLine().trim());
16         for(int i=1;i<=10;i++) {
17             System.out.println(N+ " x " + i + " = " + (N*i));
18         }
19
20
21         bufferedReader.close();
22     }
23 }
24
```

Test case 0

Test case 1

Test case 2

Compiler Message

```
Success
```

Input (stdin) — Download

```
2
```

Expected Output — Download

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
```

Q7. Java Loop II

We use the integers a, b, and n to create the following series:

$$(a + 2^0 \cdot b), (a + 2^0 \cdot b + 2^1 \cdot b), \ldots, (a + 2^0 \cdot b + 2^1 \cdot b + \ldots + 2^{n-1} \cdot b)$$

You are given q queries in the form of a, b, and n. For each query, print the series corresponding to the given a, b, and n values as a single line of n space-separated integers.

Input Format

The first line contains an integer, q, denoting the number of queries.
Each line i of the q subsequent lines contains three space-separated integers describing the respective a_i, b_i, and n_i values for that query.

Constraints

- $0 \leq q \leq 500$
- $0 \leq a, b \leq 50$
- $1 \leq n \leq 15$

Output Format

For each query, print the corresponding series on a new line. Each series must be printed in order as a single line of n space-separated integers.

Sample Input

```
2
0 2 10
5 3 5
```

Sample Output

```
2 6 14 30 62 126 254 510 1022 2046
8 14 26 50 98
```

Submitted Code

Language: Java 7 — Open in editor

```
    Scanner in = new Scanner(System.in);
    int t = in.nextInt();
    for (int i = 0; i < t; i++) {
        int a = in.nextInt();
        int b = in.nextInt();
        int n = in.nextInt();
        int sum = a;
        for (int j = 0; j < n; j++) {
            sum += b * Math.pow(2, j);
            System.out.print(sum + " ");
        }
        System.out.println();
    }
    in.close();
}
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4

Compiler Message

```
Success
```

Input (stdin) — Download

```
2
0 2 10
5 3 5
```

Expected Output — Download

```
2 6 14 30 62 126 254 510 1022 2046
8 14 26 50 98
```

Q8. Java data type

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.
```

```
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");
                //Complete the code
            }
            catch(Exception e)
            {
                System.out.println(sc.next()+" can't be fitted anywhere.");
            }

        }
    }
}
```

Q9. Java End of file

> "In computing, End Of File (commonly abbreviated EOF) is a condition in a computer operating system where no more data can be read from a data source." — (Wikipedia: End-of-file)

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.
```

Submitted Code

Language: Java 7 — Open in editor

```java
public class Solution {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int lineNumber = 1;

        while (scanner.hasNext()) {  // Keep reading until EOF
            System.out.println(lineNumber + " " + scanner.nextLine());
            lineNumber++;
        }

        scanner.close(); // Close scanner to prevent resource leaks
    }
}

```

Test case 0

Test case 1

Compiler Message

Success

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Q10. 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
```

Sample output 1

```
3
```

Submitted Code

Language: Java 7 — Open in editor

```
13
14  static {
15      Scanner sc = new Scanner(System.in);
16      B = sc.nextInt();
17      H = sc.nextInt();
18      sc.close();
19
20      if (B > 0 && H > 0) {
21          flag = true;
22      } else {
23          flag = false;
24          System.out.println("java.lang.Exception: Breadth and height must be positive");
25      }
26  }
27
```

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Input (stdin) — Download

```
1
3
```

Test case 3

Test case 4

Expected Output — Download

```
3
```

Test case 5

Test case 6

Q11. Java Int to String

You are given an integer n, you have to convert it into a string.

Please complete the partially completed code in the editor. If your code successfully converts n into a string s the code will print "Good job". Otherwise it will print "Wrong answer".

n can range between -100 to 100 inclusive.

Sample Input 0

```
100
```

Sample Output 0

```
Good job
```

Submitted Code

Language: Java 7 — Open in editor

```
13
14 String s = String.valueOf(n);
15
16   //Write your code here
17
```

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

```
Success
```

Input (stdin) — Download

```
1  100
```

Expected Output — Download

```
1  Good job
```

Q12. Java Date and Time

Function Description

Complete the findDay function in the editor below.

findDay has the following parameters:

- int: month
- int: day
- int: year

Returns

- string: the day of the week in capital letters

Input Format

A single line of input containing the space separated month, day and year, respectively, in $MM\ DD\ YYYY$ format.

Constraints

- $2000 < year < 3000$

Sample Input

```
08 05 2015
```

Sample Output

```
WEDNESDAY
```

Explanation

The day on August 5^{th} 2015 was `WEDNESDAY`.

Submitted Code

Language: Java 7 — Open in editor

```
23
24  public static String findDay(int month, int day, int year) {
25  Calendar calendar = Calendar.getInstance();
26          calendar.set(year, month - 1, day);
27
28          int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
29
30          String[] days = {"SUNDAY", "MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY", "SATURDAY"};
31
32          return days[dayOfWeek - 1];
33  }
34
35 }
36
37
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin) — Download

```
08 05 2015
```

Expected Output — Download

```
WEDNESDAY
```

Q13. Java Currency formatter

Given a double-precision number, $payment$, denoting an amount of money, use the NumberFormat class' getCurrencyInstance method to convert $payment$ into the US, Indian, Chinese, and French currency formats. Then print the formatted values as follows:

```
US: formattedPayment
India: formattedPayment
China: formattedPayment
France: formattedPayment
```

where $formattedPayment$ is $payment$ formatted according to the appropriate Locale's currency.

Note: India does not have a built-in Locale, so you must construct one where the language is `en` (i.e., English).

Input Format

A single double-precision number denoting $payment$.

Constraints

- $0 \leq payment \leq 10^9$

Output Format

On the first line, print `US: u` where u is $payment$ formatted for US currency.

On the second line, print `India: i` where i is $payment$ formatted for Indian currency.

On the third line, print `China: c` where c is $payment$ formatted for Chinese currency.

On the fourth line, print `France: f`, where f is $payment$ formatted for French currency.

Sample Input

```
12324.134
```

Sample Output

```
US: $12,324.13
India: Rs.12,324.13
China: ￥12,324.13
France: 12 324,13 €
```

Explanation

Each line contains the value of $payment$ formatted according to the four countries' respective currencies.

Submitted Code

Language: Java 7 — Open in editor

```
    // Write your code here.
    // Create Locale for India
    Locale indiaLocale = new Locale("en", "IN");

    // Create NumberFormats using Locales
    NumberFormat usFormat = NumberFormat.getCurrencyInstance(Locale.US);
    NumberFormat indiaFormat = NumberFormat.getCurrencyInstance(indiaLocale);
    NumberFormat chinaFormat = NumberFormat.getCurrencyInstance(Locale.CHINA);
    NumberFormat franceFormat = NumberFormat.getCurrencyInstance(Locale.FRANCE);

    // Format the payment amount
    String us = usFormat.format(payment);
    String india = indiaFormat.format(payment);
    String china = chinaFormat.format(payment);
    String france = franceFormat.format(payment);
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
12324.134
```

Expected Output — Download

```
US: $12,324.13
India: Rs.12,324.13
China: ￥12,324.13
France: 12 324,13 €
```

Q14. Java Strings Introduction

The elements of a String are called characters. The number of characters in a String is called the length, and it can be retrieved with the String.length() method.

Given two strings of lowercase English letters, A and B, perform the following operations:

1. Sum the lengths of A and B.
2. Determine if A is lexicographically larger than B (i.e.: does B come before A in the dictionary?).
3. Capitalize the first letter in A and B and print them on a single line, separated by a space.

Input Format

The first line contains a string A. The second line contains another string B. The strings are comprised of only lowercase English letters.

Output Format

There are three lines of output:

For the first line, sum the lengths of A and B.

For the second line, write `Yes` if A is lexicographically greater than B otherwise print `No` instead.

For the third line, capitalize the first letter in both A and B and print them on a single line, separated by a space.

Sample Input 0

```
hello
java
```

Sample Output 0

```
9
No
Hello Java
```

Explanation 0

String A is "hello" and B is "java".

A has a length of 5, and B has a length of 4; the sum of their lengths is 9.

When sorted alphabetically/lexicographically, "hello" precedes "java"; therefore, A is not greater than B and the answer is `No`.

When you capitalize the first letter of both A and B and then print them separated by a space, you get "Hello Java".

Submitted Code

Language: Java 7 — Open in editor

```
public static void main(String[] args) {

    Scanner sc=new Scanner(System.in);
    String A=sc.next();
    String B=sc.next();
    /* Enter your code here. Print output to STDOUT. */
    System.out.println(A.length() + B.length());

    // 2. Determine if A is lexicographically larger than B
    System.out.println(A.compareTo(B) > 0 ? "Yes" : "No");

    // 3. Capitalize the first letter of both strings and print them
    String capitalizedA = A.substring(0, 1).toUpperCase() + A.substring(1);
    String capitalizedB = B.substring(0, 1).toUpperCase() + B.substring(1);
    System.out.println(capitalizedA + " " + capitalizedB);
```

Test case 0

Compiler Message

```
Success
```

Test case 1 | Test case 2 | Test case 3 | Test case 4 | Test case 5

Input (stdin) — Download

```
hello
java
```

Expected Output — Download

```
9
No
Hello Java
```

Q15. Java Substring

Given a string, s, and two indices, $start$ and end, print a substring consisting of all characters in the inclusive range from $start$ to $end - 1$. You'll find the String class' substring method helpful in completing this challenge.

Input Format

The first line contains a single string denoting s.
The second line contains two space-separated integers denoting the respective values of $start$ and end.

Constraints

- $1 \leq |s| \leq 100$
- $0 \leq start < end \leq n$
- String s consists of English alphabetic letters (i.e., $[a - zA - Z]$) only.

Output Format

Print the substring in the inclusive range from $start$ to $end - 1$.

Sample Input

```
Helloworld
3 7
```

Sample Output

```
lowo
```

Submitted Code

Language: Java 7 — Open in editor

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

public class Solution {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String S = in.next();
        int start = in.nextInt();
        int end = in.nextInt();
        System.out.println(S.substring(start, end));
    }
}
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5

Compiler Message

```
Success
```

Input (stdin) — Download

```
Helloworld
3 7
```

Expected Output — Download

```
lowo
```

Q16. Java Substring Comparisons

We define the following terms:

- Lexicographical Order, also known as alphabetic or dictionary order, orders characters as follows:

$$A < B < \ldots < Y < Z < a < b < \ldots < y < z$$

  For example, `ball < cat`, `dog < dorm`, `Happy < happy`, `Zoo < ball`.
- A substring of a string is a contiguous block of characters in the string. For example, the substrings of `abc` are `a`, `b`, `c`, `ab`, `bc`, and `abc`.

Given a string, s, and an integer, k, complete the function so that it finds the lexicographically smallest and largest substrings of length k.

Function Description

Complete the getSmallestAndLargest function in the editor below.

getSmallestAndLargest has the following parameters:

- string s: a string
- int k: the length of the substrings to find

Returns

- string: the string ' + "\n" + ' where and are the two substrings

Input Format

The first line contains a string denoting s.
The second line contains an integer denoting k.

Constraints

- $1 \leq |s| \leq 1000$
- s consists of English alphabetic letters only (i.e., `[a-zA-Z]`).

Sample Input 0

```
welcometojava
3
```

Sample Output 0

```
ava
wel
```

Explanation 0

String $s =$ `"welcometojava"` has the following lexicographically-ordered substrings of length $k = 3$:

```
["ava", "com", "elc", "eto", "jav", "lco", "met", "oja", "ome", "toj", "wel"]
```

We then return the first (lexicographically smallest) substring and the last (lexicographically largest) substring as two newline-separated values (i.e., `ava\nwel`).

The stub code in the editor then prints `ava` as our first line of output and `wel` as our second line of output.

Submitted Code

Language: Java 7 — Open in editor

```
    String largest = s.substring(0, k);

    for (int i = 1; i <= s.length() - k; i++) {
        String sub = s.substring(i, i + k);
        if (sub.compareTo(smallest) < 0) {
            smallest = sub;
        }
        if (sub.compareTo(largest) > 0) {
            largest = sub;
        }
    }
    return smallest + "\n" + largest;
}
```

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Input (stdin) — Download

```
welcometojava
3
```

Test case 3

Test case 4

Expected Output — Download

```
ava
wel
```

Test case 5

Test case 6

Q17. Java String Reverse

A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward or forward.

Given a string A, print `Yes` if it is a palindrome, print `No` otherwise.

Constraints

- A will consist at most 50 lower case english letters.

Sample Input

```
madam
```

Sample Output

```
Yes
```

Submitted Code

Language: Java 7 — Open in editor

```
        Scanner sc=new Scanner(System.in);
        String A=sc.next();
        /* Enter your code here. Print output to STDOUT. */
              sc.close();

        // Check if the string is a palindrome
        String reversed = new StringBuilder(A).reverse().toString();
        if (A.equals(reversed)) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }
    }
}
```

Q18. Java Anagrams

Two strings, a and b, are called anagrams if they contain all the same characters in the same frequencies. For this challenge, the test is not case-sensitive. For example, the anagrams of `CAT` are `CAT`, `ACT`, `tac`, `TCA`, `aTC`, and `CtA`.

Function Description

Complete the isAnagram function in the editor.

isAnagram has the following parameters:

- string a: the first string
- string b: the second string

Returns

- boolean: If a and b are case-insensitive anagrams, return true. Otherwise, return false.

Input Format

The first line contains a string a.

The second line contains a string b.

Constraints

- $1 \leq length(a), length(b) \leq 50$
- Strings a and b consist of English alphabetic characters.
- The comparison should NOT be case sensitive.

Sample Input 0

```
anagram
margana
```

Sample Output 0

```
Anagrams
```

Submitted Code

Language: Java 7 — Open in editor

```
static boolean isAnagram(String a, String b) {
    // Complete the function
    if (a.length() != b.length()) {
        return false;
    }

    char[] arrA = a.toLowerCase().toCharArray();
    char[] arrB = b.toLowerCase().toCharArray();
    java.util.Arrays.sort(arrA);
    java.util.Arrays.sort(arrB);

    return java.util.Arrays.equals(arrA, arrB);
}
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
anagram
margana
```

Expected Output — Download

```
Anagrams
```

Q19. Java String tokens

Given a string, s, matching the regular expression `[A-Za-z !,?._'@]+`, split the string into tokens. We define a token to be one or more consecutive English alphabetic letters. Then, print the number of tokens, followed by each token on a new line.

Note: You may find the String.split method helpful in completing this challenge.

Input Format

A single string, s.

Constraints

- $1 \leq$ length of $s \leq 4 \cdot 10^5$
- s is composed of any of the following: English alphabetic letters, blank spaces, exclamation points (!), commas (,), question marks (?), periods (.), underscores (_), apostrophes ('), and at symbols (@).

Output Format

On the first line, print an integer, n, denoting the number of tokens in string s (they do not need to be unique). Next, print each of the n tokens on a new line in the same order as they appear in input string s.

Sample Input

```
He is a very very good boy, isn't he?
```

Sample Output

```
10
He
is
a
very
very
good
boy
isn
t
he
```

Submitted Code

Language: Java 7 — Open in editor

```java
public class Solution {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String s = scan.nextLine();
        // Write your code here.
        String[] tokens = s.trim().split("[^a-zA-Z]+");

        // Handling the case where input is empty or only has special characters
        if (tokens.length == 1 && tokens[0].isEmpty()) {
            System.out.println(0);
        } else {
            System.out.println(tokens.length);
            for (String token : tokens) {
                System.out.println(token);
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
He is a very very good boy, isn't he?
```

Expected Output — Download

```
10
He
is
a
very
very
```

Q20.Pattern Syntax Checker

Using **Regex**, we can easily match or search for patterns in a text. Before searching for a pattern, we have to specify one using some well-defined syntax.

In this problem, you are given a pattern. You have to check whether the syntax of the given pattern is valid.

Note: In this problem, a regex is only valid if you can compile it using the Pattern.compile method.

Input Format

The first line of input contains an integer N, denoting the number of test cases. The next N lines contain a string of any printable characters representing the pattern of a regex.

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
```

Submitted Code

Language: Java 7 — Open in editor

```
{
    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();
            //Write your code
            try {
                // Try to compile the pattern
                Pattern.compile(pattern);
                System.out.println("Valid");  // If no exception occurs, it's a valid regex
            } catch (PatternSyntaxException e) {
                System.out.println("Invalid");  // If a PatternSyntaxException is thrown, it's invalid
            }
            testCases--;
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

Input (stdin) — Download

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

Expected Output — Download

```
Valid
Invalid
Invalid
```

Q21.Java Regex

Write a class called MyRegex which will contain a string pattern. You need to write a regular expression and assign it to the pattern such that it can be used to validate an IP address. Use the following definition of an IP address:

```
IP address is a string in the form "A.B.C.D", where the value of A, B, C, and D may range from 0
```

Some valid IP address:

```
000.12.12.034
121.234.12.12
23.45.12.56
```

Some invalid IP address:

```
000.12.234.23.23
666.666.23.23
.213.123.23.32
23.45.22.32.
I.Am.not.an.ip
```

In this problem you will be provided strings containing any combination of ASCII characters. You have to write a regular expression to find the valid IPs.

Just write the MyRegex class which contains a String $pattern$. The string should contain the correct regular expression.

(MyRegex class MUST NOT be public)

Sample Input

```
000.12.12.034
121.234.12.12
23.45.12.56
00.12.123.123123.123
122.23
Hello.IP
```

Submitted Code

Language: Java 7 — Open in editor

```
16
17
18 //Write your code here
19 class MyRegex {
20     // Write your code here
21     String pattern = "^([01]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])\\.([01]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])\\.([01]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])\\.
   ([01]?[0-9]{1,2}|2[0-4][0-9]|25[0-5])$";
22 }
23
24
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
1 000.12.12.034
2 121.234.12.12
3 23.45.12.56
4 00.12.123.123123.123
5 122.23
```

Q22. Java Regex 2 - Duplicate Words

In this challenge, we use regular expressions (RegEx) to remove instances of words that are repeated more than once, but retain the first occurrence of any case-insensitive repeated word. For example, the words `love` and `to` are repeated in the sentence `I love Love to To tO code`. Can you complete the code in the editor so it will turn `I love Love to To tO code` into `I love to code`?

To solve this challenge, complete the following three lines:

1. Write a RegEx that will match any repeated word.
2. Complete the second compile argument so that the compiled RegEx is case-insensitive.
3. Write the two necessary arguments for replaceAll such that each repeated word is replaced with the very first instance the word found in the sentence. It must be the exact first occurrence of the word, as the expected output is case-sensitive.

Note: This challenge uses a custom checker; you will fail the challenge if you modify anything other than the three locations that the comments direct you to complete. To restore the editor's original stub code, create a new buffer by clicking on the branch icon in the top left of the editor.

Input Format

The following input is handled for you the given stub code:

The first line contains an integer, n, denoting the number of sentences.
Each of the n subsequent lines contains a single sentence consisting of English alphabetic letters and whitespace characters.

Constraints

- Each sentence consists of at most 10^4 English alphabetic letters and whitespaces.
- $1 \leq n \leq 100$

Output Format

Stub code in the editor prints the sentence modified by the replaceAll line to stdout. The modified string must be a modified version of the initial sentence where all repeat occurrences of each word are removed.

Submitted Code

Language: Java 7 — Open in editor

```
    String regex = "\\b(\\w+)(?:\\s+\\1\\b)+";
    Pattern p = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);

    Scanner in = new Scanner(System.in);
    int numSentences = Integer.parseInt(in.nextLine());

    while (numSentences-- > 0) {
        String input = in.nextLine();

        Matcher m = p.matcher(input);

        // Check for subsequences of input that match the compiled pattern
        while (m.find()) {
            input = input.replaceAll(m.group(), m.group(1));
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

Input (stdin) — Download

```
5
Goodbye bye bye world world world
Sam went went to to to his business
Reya is is the the best player in eye eye game
in inthe
Hello hello Ab aB
```

Expected Output — Download

```
Goodbye bye world
```

Q23. Valid Username Regular Expression

You are updating the username policy on your company's internal networking platform. According to the policy, a username is considered valid if all the following constraints are satisfied:

- The username consists of 8 to 30 characters inclusive. If the username consists of less than 8 or greater than 30 characters, then it is an invalid username.
- The username can only contain alphanumeric characters and underscores (_). Alphanumeric characters describe the character set consisting of lowercase characters $[a-z]$, uppercase characters $[A-Z]$, and digits $[0-9]$.
- The first character of the username must be an alphabetic character, i.e., either lowercase character $[a-z]$ or uppercase character $[A-Z]$.

For example:

Username	Validity
Julia	INVALID; Username length < 8 characters
Samantha	VALID
Samantha_21	VALID
1Samantha	INVALID; Username begins with non-alphabetic character
Samantha?10_2A	INVALID; '?' character not allowed

Update the value of regularExpression field in the UsernameValidator class so that the regular expression only matches with valid usernames.

Input Format

The first line of input contains an integer n, describing the total number of usernames. Each of the next n lines contains a string describing the username. The locked stub code reads the inputs and validates the username.

Constraints

- $1 \leq n \leq 100$
- The username consists of any printable characters.

Output Format

For each of the usernames, the locked stub code prints `Valid` if the username is valid; otherwise `Invalid` each on a new line.

Submitted Code

Language: Java 7 — Open in editor

```
class UsernameValidator {
    /*
     * Write regular expression here.
     */
    public static final String regularExpression = "^[a-zA-Z][a-zA-Z0-9_]{7,29}$";
}
```

Test case 0 — Success

Test case 1 — Input (stdin) — Download

```
8
Julia
Samantha
Samantha_21
1Samantha
Samantha?10_2A
JuliaZ007
Julia@007
_Julia007
```

Test case 2 — Test case 3 — Test case 4 — Test case 5 — Test case 6

Q24. Tag Content Extractor

In a tag-based language like XML or HTML, contents are enclosed between a start tag and an end tag like `<tag>contents</tag>`. Note that the corresponding end tag starts with a `/`.

Given a string of text in a tag-based language, parse this text and retrieve the contents enclosed within sequences of well-organized tags meeting the following criterion:

1. The name of the start and end tags must be same. The HTML code `<h1>Hello World</h2>` is not valid, because the text starts with an `h1` tag and ends with a non-matching `h2` tag.
2. Tags can be nested, but content between nested tags is considered not valid. For example, in `<h1><a>contents</a>invalid</h1>`, `contents` is valid but `invalid` is not valid.
3. Tags can consist of any printable characters.

Input Format

The first line of input contains a single integer, N (the number of lines).
The N subsequent lines each contain a line of text.

Constraints

- $1 \leq N \leq 100$
- Each line contains a maximum of 10^4 printable characters.
- The total number of characters in all test cases will not exceed 10^6.

Output Format

For each line, print the content enclosed within valid tags.
If a line contains multiple instances of valid content, print out each instance of valid content on a new line; if no valid content is found, print `None`.

Sample Input

```
4
<h1>Nayeem loves counseling</h1>
<h1><h1>Sanjay has no watch</h1></h1><par>So wait for a while</par>
<Amee>safat codes like a ninja</amee>
<SA premium>Imtiaz has a secret crush</SA premium>
```

Submitted Code

Language: Java 7 — Open in editor

```
        while(testCases>0){
            String line = in.nextLine();

            //Write your code here
             //Write your code here
            boolean found = false;
            Pattern pattern = Pattern.compile("<(.+?)>([^<]+)</\\1>");
            Matcher matcher = pattern.matcher(line);

            while(matcher.find()){
                System.out.println(matcher.group(2));
                found = true;
            }
            if(!found){
                System.out.println("None");
            }
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
4
<h1>Nayeem loves counseling</h1>
<h1><h1>Sanjay has no watch</h1></h1><par>So wait for a while<par>
<Amee>safat codes like a ninja</amee>
<SA premium>Imtiaz has a secret crush</SA premium>
```

Expected Output — Download

```
Nayeem loves counseling
Sanjay has no watch
```

Q25. 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.

Sample Input

```
9
-100
50
0
56.6
90
0.12
.12
02.34
000.000
```

Submitted Code

Language: Java 7 — Open in editor

```
14
15
16      Arrays.sort(s, 0, n, new Comparator<String>() {
17          public int compare(String a, String b) {
18              BigDecimal aVal = new BigDecimal(a);
19              BigDecimal bVal = new BigDecimal(b);
20              return bVal.compareTo(aVal); // Descending order
21          }
22      });
23
24
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
9
-100
50
0
56.6
90
0.12
.12
02.34
```

Q26. Java Primality Test

A prime number is a natural number greater than 1 whose only positive divisors are 1 and itself. For example, the first six prime numbers are 2, 3, 5, 7, 11, and 13.

Given a large integer, n, use the Java BigInteger class' isProbablePrime method to determine and print whether it's `prime` or `not prime`.

Input Format

A single line containing an integer, n (the number to be checked).

Constraints

- n contains at most 100 digits.

Output Format

If n is a prime number, print `prime`; otherwise, print `not prime`.

Sample Input

```
13
```

Sample Output

```
prime
```

Explanation

The only positive divisors of 13 are 1 and 13, so we print `prime`.

Submitted Code

Language: Java 7 — Open in editor

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


public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

        String n = bufferedReader.readLine();
```

Q27. 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
```

Sample Output

```
1254
24680
```

Explanation

$1234 + 20 = 1254$

$1234 \times 20 = 24680$

Submitted Code

Language: Java 7 — Open in editor

```
    // Read the two big integers as strings from input
    String num1 = scanner.nextLine();
    String num2 = scanner.nextLine();

    // Convert strings to BigInteger
    BigInteger a = new BigInteger(num1);
    BigInteger b = new BigInteger(num2);

    // Perform addition and multiplication
    BigInteger sum = a.add(b);
    BigInteger product = a.multiply(b);

    // Print the results
    System.out.println(sum);
    System.out.println(product);
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
1234
20
```

Expected Output — Download

```
1254
24680
```

Q28. Java 1D Array

Task

The code in your editor does the following:

1. Reads an integer from stdin and saves it to a variable, n, denoting some number of integers.
2. Reads n integers corresponding to $a_0, a_1, \ldots, a_{n-1}$ from stdin and saves each integer a_i to a variable, val.
3. Attempts to print each element of an array of integers named a.

Write the following code in the unlocked portion of your editor:

1. Create an array, a, capable of holding n integers.
2. Modify the code in the loop so that it saves each sequential value to its corresponding location in the array. For example, the first value must be stored in a_0, the second value must be stored in a_1, and so on.

Good luck!

Input Format

The first line contains a single integer, n, denoting the size of the array.

Each line i of the n subsequent lines contains a single integer denoting the value of element a_i.

Output Format

You are not responsible for printing any output to stdout. Locked code in the editor loops through array a and prints each sequential element on a new line.

Sample Input

```
5
10
20
30
40
50
```

Submitted Code

Language: Java 7 — Open in editor

```
10
11 int[] a = new int[n];   // Create an array of size n
12
13       // Read n integers and store them in the array
14       for (int i = 0; i < n; i++) {
15           a[i] = scan.nextInt();
16       }
17
18
```

Test case 0

Test case 1

Compiler Message

Success

Input (stdin) — Download

```
5
10
20
30
40
50
```

Expected Output — Download

Q29. Java 2D Array

You are given a $6 * 6$ 2D array. An hourglass in an array is a portion shaped like this:

```
a b c
  d
e f g
```

For example, if we create an hourglass using the number 1 within an array full of zeros, it may look like this:

```
1 1 1 0 0 0
0 1 0 0 0 0
1 1 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Actually, there are many hourglasses in the array above. The three leftmost hourglasses are the following:

```
1 1 1    1 1 0    1 0 0
  1        0        0
1 1 1    1 1 0    1 0 0
```

The sum of an hourglass is the sum of all the numbers within it. The sum for the hourglasses above are 7, 4, and 2, respectively.

In this problem you have to print the largest sum among all the hourglasses in the array.

Input Format

There will be exactly 6 lines, each containing 6 integers seperated by spaces. Each integer will be between -9 and 9 inclusive.

Output Format

Print the answer to this problem on a single line.

Submitted Code

Language: Java 7

```
23
24      // Initialize maxSum to the smallest possible hourglass sum
25      int maxSum = Integer.MIN_VALUE;
26
27      // Loop to go through each possible hourglass center
28      for (int i = 1; i < 5; i++) {
29          for (int j = 1; j < 5; j++) {
30              int sum = arr.get(i - 1).get(j - 1) + arr.get(i - 1).get(j) + arr.get(i - 1).get(j + 1)
31                      + arr.get(i).get(j)
32                      + arr.get(i + 1).get(j - 1) + arr.get(i + 1).get(j) + arr.get(i + 1).get(j + 1);
33
34              maxSum = Math.max(maxSum, sum);
35          }
36      }
37
38      // Print the maximum hourglass sum
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Q30. Java Subarray

We define the following:

- A subarray of an n-element array is an array composed from a contiguous block of the original array's elements. For example, if $array = [1, 2, 3]$, then the subarrays are $[1]$, $[2]$, $[3]$, $[1, 2]$, $[2, 3]$, and $[1, 2, 3]$. Something like $[1, 3]$ would not be a subarray as it's not a contiguous subsection of the original array.
- The sum of an array is the total sum of its elements.
  - An array's sum is negative if the total sum of its elements is negative.
  - An array's sum is positive if the total sum of its elements is positive.

Given an array of n integers, find and print its number of negative subarrays on a new line.

Input Format

The first line contains a single integer, n, denoting the length of array $A = [a_0, a_1, \ldots, a_{n-1}]$.

The second line contains n space-separated integers describing each respective element, a_i, in array A.

Constraints

- $1 \leq n \leq 100$
- $-10^4 \leq a_i \leq 10^4$

Output Format

Print the number of subarrays of A having negative sums.

Sample Input

```
5
1 -2 4 -5 1
```

Sample Output

```
9
```

Submitted Code

Language: Java 7 — Open in editor

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

public class Solution {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt(); // Read array size
        int[] arr = new int[n];    // Initialize array

        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt(); // Read array elements
        }
```

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Input (stdin) — Download

```
5
1 -2 4 -5 1
```

Test case 3

Test case 4

Expected Output — Download

```
9
```

Test case 5

Q31. Java Arraylist

Sometimes it's better to use dynamic size arrays. Java's Arraylist can provide you this feature. Try to solve this problem using Arraylist.

You are given n lines. In each line there are zero or more integers. You need to answer a few queries where you need to tell the number located in y^{th} position of x^{th} line.

Take your input from System.in.

Input Format

The first line has an integer n. In each of the next n lines there will be an integer d denoting number of integers on that line and then there will be d space-separated integers. In the next line there will be an integer q denoting number of queries. Each query will consist of two integers x and y.

Constraints

- $1 <= n <= 20000$
- $0 <= d <= 50000$
- $1 <= q <= 1000$
- $1 <= x <= n$

Each number will fit in signed integer.

Total number of integers in n lines will not cross 10^5.

Output Format

In each line, output the number located in y^{th} position of x^{th} line. If there is no such position, just print "ERROR!"

Submitted Code

Language: Java 7 — Open in editor

```
22
23      int q = scanner.nextInt(); // number of queries
24
25      for (int i = 0; i < q; i++) {
26          int x = scanner.nextInt(); // line number (1-based)
27          int y = scanner.nextInt(); // position in line (1-based)
28
29          // Adjust to 0-based index and check bounds
30          if (x <= listOfLists.size() && x > 0) {
31              List<Integer> line = listOfLists.get(x - 1);
32
33              if (y <= line.size() && y > 0) {
34                  System.out.println(line.get(y - 1));
35              } else {
36                  System.out.println("ERROR!");
37              }
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin) — Download

```
5
5 41 77 74 22 44
1 12
4 37 34 36 52
0
3 20 22 33
5
1 3
3 4
```

Q32. Java 1D Array (Part 2)

Let's play a game on an array! You're standing at index 0 of an n-element array named $game$. From some index i (where $0 \leq i < n$), you can perform one of the following moves:

- Move Backward: If cell $i-1$ exists and contains a 0, you can walk back to cell $i-1$.
- Move Forward:
  - If cell $i+1$ contains a zero, you can walk to cell $i+1$.
  - If cell $i+leap$ contains a zero, you can jump to cell $i+leap$.
  - If you're standing in cell $n-1$ or the value of $i+leap \geq n$, you can walk or jump off the end of the array and win the game.

In other words, you can move from index i to index $i+1$, $i-1$, or $i+leap$ as long as the destination index is a cell containing a 0. If the destination index is greater than $n-1$, you win the game.

Function Description

Complete the canWin function in the editor below.

canWin has the following parameters:

- int leap: the size of the leap
- int game[n]: the array to traverse

Returns

- boolean: true if the game can be won, otherwise false

Input Format

The first line contains an integer, q, denoting the number of queries (i.e., function calls).

The $2 \cdot q$ subsequent lines describe each query over two lines:

1. The first line contains two space-separated integers describing the respective values of n and $leap$.
2. The second line contains n space-separated binary integers (i.e., zeroes and ones) describing the respective values of $game_0, game_1, \ldots, game_{n-1}$.

Constraints

- $1 \leq q \leq 5000$
- $2 \leq n \leq 100$
- $0 \leq leap \leq 100$
- It is guaranteed that the value of $game[0]$ is always 0.

Submitted Code

Language: Java 7 — Open in editor

```
8
9   // Recursive DFS with visited array
10  private static boolean canWinFrom(int i, int leap, int[] game, boolean[] visited) {
11      // Win condition: you're out of bounds
12      if (i >= game.length) return true;
13
14      // Invalid move: index out of range, already visited, or blocked
15      if (i < 0 || game[i] == 1 || visited[i]) return false;
16
17      // Mark as visited
18      visited[i] = true;
19
20      // Try all possible moves
21      return canWinFrom(i + leap, leap, game, visited) ||
22          canWinFrom(i + 1, leap, game, visited) ||
23          canWinFrom(i - 1, leap, game, visited);
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
4
5 3
0 0 0 0 0
6 5
0 0 0 1 1 1
6 3
0 0 1 1 1 0
3 1
0 1 0
```

Q33. Java List

For this problem, we have 2 types of queries you can perform on a List:

1. Insert y at index x:

```
Insert
x y
```

2. Delete the element at index x:

```
Delete
x
```

Given a list, L, of N integers, perform Q queries on the list. Once all queries are completed, print the modified list as a single line of space-separated integers.

Input Format

The first line contains an integer, N (the initial number of elements in L).

The second line contains N space-separated integers describing L.

The third line contains an integer, Q (the number of queries).

The $2Q$ subsequent lines describe the queries, and each query is described over two lines:

- If the first line of a query contains the String **Insert**, then the second line contains two space separated integers x y, and the value y must be inserted into L at index x.
- If the first line of a query contains the String **Delete**, then the second line contains index x, whose element must be deleted from L.

Constraints

- $1 \leq N \leq 4000$
- $1 \leq Q \leq 4000$
- Each element in is a 32-bit integer.

Output Format

Print the updated list L as a single line of space-separated integers.

Submitted Code

Language: Java 7

```
    int q = scan.nextInt(); // number of queries

    // Process each query
    for (int i = 0; i < q; i++) {
        String action = scan.next(); // Read the command: "Insert" or "Delete"

        if (action.equals("Insert")) {
            int index = scan.nextInt();
            int value = scan.nextInt();
            list.add(index, value); // Insert value at specified index
        } else if (action.equals("Delete")) {
            int index = scan.nextInt();
            list.remove(index); // Remove element at specified index
        }
    }
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Q34. Java Map

You are given a phone book that consists of people's names and their phone number. After that you will be given some person's name as query. For each query, print the phone number of that person.

Input Format

The first line will have an integer n denoting the number of entries in the phone book. Each entry consists of two lines: a name and the corresponding phone number.

After these, there will be some queries. Each query will contain a person's name. Read the queries until end-of-file.

Constraints:

A person's name consists of only lower-case English letters and it may be in the format 'first-name last-name' or in the format 'first-name'. Each phone number has exactly 8 digits without any leading zeros.

$1 \leq n \leq 100000$

$1 \leq Query \leq 100000$

Output Format

For each case, print "Not found" if the person has no entry in the phone book. Otherwise, print the person's name and phone number. See sample output for the exact format.

To make the problem easier, we provided a portion of the code in the editor. You can either complete that code or write completely on your own.

Sample Input

```
3
uncle sam
99912222
tom
11122222
harry
12299933
uncle sam
uncle tom
harry
```

Submitted Code

Language: Java 7 — Open in editor

```
    // Create the phone book map
    Map<String, Integer> phoneBook = new HashMap<>();

    // Read and store the phone book entries
    for(int i = 0; i < n; i++) {
        String name = in.nextLine();
        int phone = in.nextInt();
        in.nextLine(); // consume the newline after the phone number
        phoneBook.put(name, phone);
    }

    // Process the queries until end-of-file
    while(in.hasNext()) {
        String query = in.nextLine();
        if(phoneBook.containsKey(query)) {
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Input (stdin) — Download

```
3
uncle sam
99912222
tom
11122222
harry
12299933
uncle sam
uncle tom
```

Q35. Java Stack

```
In computer science, a stack or LIFO (last in, first out) is an abstract data type that serves as a
```

A string containing only parentheses is balanced if the following is true: 1. if it is an empty string 2. if A and B are correct, AB is correct, 3. if A is correct, (A) and {A} and [A] are also correct.

Examples of some correctly balanced strings are: "{}()", "[{()}]", "({()})"

Examples of some unbalanced strings are: "{()", "({)}", "[[", "}{" etc.

Given a string, determine if it is balanced or not.

Input Format

There will be multiple lines in the input file, each having a single non-empty string. You should read input till end-of-file.

The part of the code that handles input operation is already provided in the editor.

Output Format

For each case, print 'true' if the string is balanced, 'false' otherwise.

Sample Input

```
{}()
({()})
{}(
[]
```

Sample Output

```
true
true
false
true
```

Submitted Code

Language: Java 7 — Open in editor

```
    Stack<Character> stack = new Stack<>();

    for (char ch : input.toCharArray()) {
        if (ch == '(' || ch == '{' || ch == '[') {
            stack.push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (stack.isEmpty()) return false;
            char top = stack.pop();
            if (!matches(top, ch)) return false;
        }
    }

    return stack.isEmpty(); // All open brackets must be matched
}
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
{}()
({()})
{}(
```

Q36. Java Generics

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.

Submitted Code

Language: Java 7 — Open in editor

```
class Printer
{
   //Write your code here
  <T> void printArray(T[] array) {
        for (T element : array)
            System.out.println(element);
    }


}
```

Test case 0

Compiler Message

Success

Expected Output — Download

```
1
2
3
Hello
World
```

Q37. Java Comparator

Comparators are used to compare two objects. In this challenge, you'll create a comparator and use it to sort an array.

The Player class is provided for you in your editor. It has 2 fields: a $name$ String and a $score$ integer.

Given an array of n Player objects, write a comparator that sorts them in order of decreasing score; if 2 or more players have the same score, sort those players alphabetically by name. To do this, you must create a Checker class that implements the Comparator interface, then write an int compare(Player a, Player b) method implementing the Comparator.compare(T o1, T o2) method.

Input Format

Input from stdin is handled by the locked stub code in the Solution class.

The first line contains an integer, n, denoting the number of players.

Each of the n subsequent lines contains a player's $name$ and $score$, respectively.

Constraints

- $0 \leq score \leq 1000$
- 2 players can have the same name.
- Player names consist of lowercase English letters.

Output Format

You are not responsible for printing any output to stdout. The locked stub code in Solution will create a Checker object, use it to sort the Player array, and print each sorted element.

Submitted Code

Language: Java 7 — Open in edit

```
// Write your Checker class here

class Checker implements Comparator<Player> {
    @Override
    public int compare(Player a, Player b) {
        // Compare by score in descending order
        if (a.score != b.score) {
            return b.score - a.score;
        }
        // Compare by name in alphabetical order if scores are equal
        return a.name.compareTo(b.name);
    }
}
```

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Input (stdin) — Download

```
5
amy 100
david 100
heraldo 50
aakansha 75
aleksa 150
```

Test case 3

Test case 4

Test case 5

Q38. Java Sort

You are given a list of student information: ID, FirstName, and CGPA. Your task is to rearrange them according to their CGPA in decreasing order. If two student have the same CGPA, then arrange them according to their first name in alphabetical order. If those two students also have the same first name, then order them according to their ID. No two students have the same ID.

Hint: You can use comparators to sort a list of objects. See the oracle docs to learn about comparators.

Input Format

The first line of input contains an integer N, representing the total number of students. The next N lines contains a list of student information in the following structure:

```
ID Name CGPA
```

Constraints

$2 \leq N \leq 1000$

$0 \leq ID \leq 100000$

$5 \leq |Name| \leq 30$

$0 \leq CGPA \leq 4.00$

The name contains only lowercase English letters. The ID contains only integer numbers without leading zeros. The CGPA will contain, at most, 2 digits after the decimal point.

Output Format

After rearranging the students according to the above rules, print the first name of each student on a separate line.

Submitted Code

Language: Java 7 Open in edit

```
        super();
        this.id = id;
        this.fname = fname;
        this.cgpa = cgpa;
    }
    public int getId() {
        return id;
    }
    public String getFname() {
        return fname;
    }
    public double getCgpa() {
        return cgpa;
    }
}
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
5
33 Rumpa 3.68
85 Ashis 3.85
56 Samiha 3.75
19 Samara 3.75
22 Fahim 3.76
```

Expected Output Download

Q39. Java Dequeue

In computer science, a double-ended queue (dequeue, often abbreviated to deque, pronounced deck) is an abstract data type that generalizes a queue, for which elements can be added to or removed from either the front (head) or back (tail).

Deque interfaces can be implemented using various types of collections such as `LinkedList` or `ArrayDeque` classes. For example, deque can be declared as:

```
Deque deque = new LinkedList<>();
or
Deque deque = new ArrayDeque<>();
```

You can find more details about Deque here.

In this problem, you are given N integers. You need to find the maximum number of unique integers among all the possible contiguous subarrays of size M.

Note: Time limit is 3 second for this problem.

Input Format

The first line of input contains two integers N and M: representing the total number of integers and the size of the subarray, respectively. The next line contains N space separated integers.

Constraints

$1 \leq N \leq 100000$

$1 \leq M \leq 100000$

$M \leq N$

The numbers in the array will range between $[0, 10000000]$.

Output Format

Print the maximum number of unique integers among all possible contiguous subarrays of size M.

Submitted Code

Language: Java 7 — Open in editor

```
        // Add the current number to the deque and the set
        deque.add(num);
        set.add(num);

        // If the deque size exceeds the subarray size (m), remove the oldest element
        if (deque.size() > m) {
            int removed = deque.poll();
            // If the removed element is no longer in the deque, remove it from the set
            if (!deque.contains(removed)) {
                set.remove(removed);
            }
        }

        // Update the maximum unique count
```

Test case 0

Compiler Message

Success

Input (stdin) — Download

```
6 3
5 3 5 2 3 2
```

Expected Output — Download

```
3
```

Test case 1 🔒

Test case 2 🔒

Test case 3 🔒

Test case 4 🔒

Test case 5 🔒

Test case 6 🔒

Q40. Java BitSet

Java's BitSet class implements a vector of bit values (i.e.: $false$ (0) or $true$ (1)) that grows as needed, allowing us to easily manipulate bits while optimizing space (when compared to other collections). Any element having a bit value of 1 is called a set bit.

Given 2 BitSets, B_1 and B_2, of size N where all bits in both BitSets are initialized to 0, perform a series of M operations. After each operation, print the number of set bits in the respective BitSets as two space-separated integers on a new line.

Input Format

The first line contains 2 space-separated integers, N (the length of both BitSets B_1 and B_2) and M (the number of operations to perform), respectively.

The M subsequent lines each contain an operation in one of the following forms:

- AND $<\text{set}>$ $<\text{set}>$
- OR $<\text{set}>$ $<\text{set}>$
- XOR $<\text{set}>$ $<\text{set}>$
- FLIP $<\text{set}>$ $<\text{index}>$
- SET $<\text{set}>$ $<\text{index}>$

In the list above, $<\text{set}>$ is the integer 1 or 2, where 1 denotes B_1 and 2 denotes B_2.

$<\text{index}>$ is an integer denoting a bit's index in the BitSet corresponding to $<\text{set}>$.

For the binary operations AND, OR, and XOR, operands are read from left to right and the BitSet resulting from the operation replaces the contents of the first operand. For example:

```
AND 2 1
```

B_2 is the left operand, and B_1 is the right operand. This operation should assign the result of $B_2 \wedge B_1$ to B_2.

Constraints

- $1 \leq N \leq 1000$
- $1 \leq M \leq 10000$

Output Format

After each operation, print the respective number of set bits in BitSet B_1 and BitSet B_2 as 2 space-separated integers on a new line.

Submitted Code

Language: Java 7

```
15
16      BitSet bitSet1 = new BitSet(n); // BitSet 1
17      BitSet bitSet2 = new BitSet(n); // BitSet 2
18
19      for (int i = 0; i < m; i++) {
20          String operation = in.next(); // Operation type
21          int x = in.nextInt(); // BitSet number
22          int y = in.nextInt(); // Operand or index
23
24          switch (operation) {
25              case "AND":
26                  if (x == 1) {
27                      bitSet1.and(bitSet2);
28                  } else {
29                      bitSet2.and(bitSet1);
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin) Download

```
5 4
AND 1 2
SET 1 4
FLIP 2 2
OR 2 1
```

Expected Output Download

```
0 0
1 0
```

Q41. Java Inheritance I

Using inheritance, one class can acquire the properties of others. Consider the following Animal class:

```
class Animal{
    void walk(){
        System.out.println("I am walking");
    }
}
```

This class has only one method, walk. Next, we want to create a Bird class that also has a fly method. We do this using extends keyword:

```
class Bird extends Animal {
    void fly() {
        System.out.println("I am flying");
    }
}
```

Finally, we can create a Bird object that can both fly and walk.

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

        Bird bird = new Bird();
        bird.walk();
        bird.fly();
    }
}
```

The above code will print:

```
I am walking
I am flying
```

This means that a Bird object has all the properties that an Animal object has, as well as some additional unique properties.

The code above is provided for you in your editor. You must add a sing method to the Bird class, then modify the main method accordingly so that the code prints the following lines:

Submitted Code

Language: Java 7 — Open in editor

```
12
13
14 class Bird extends Animal{
15     void fly(){
16         System.out.println("I am flying");
17     }
18     void sing(){
19         System.out.println("I am singing");
20     }
21 }
22
23
```

Test case 0

Compiler Message

```
Success
```

Expected Output — Download

```
1 I am walking
2 I am flying
3 I am singing
```

Q42. Java Inheritance II

Write the following code in your editor below:

1. A class named Arithmetic with a method named add that takes 2 integers as parameters and returns an integer denoting their sum.
2. A class named Adder that inherits from a superclass named Arithmetic.

Your classes should not be be $\mathbf{public}$.

Input Format

You are not responsible for reading any input from stdin; a locked code stub will test your submission by calling the add method on an Adder object and passing it 2 integer parameters.

Output Format

You are not responsible for printing anything to stdout. Your add method must return the sum of its parameters.

Sample Output

The main method in the Solution class above should print the following:

```
My superclass is: Arithmetic
42 13 20
```

Submitted Code

Language: Java 7 — Open in editor

```
6
7
8  //Write your code here
9  class Arithmetic{
10     int add(int a, int b){
11         return a+b;
12     }
13 }
14
15 class Adder extends Arithmetic{}
16
```

Test case 0

Compiler Message

Success

Expected Output — Download

```
1  My superclass is: Arithmetic
2  42 13 20
```

Q43. Java Abstract Class

A Java abstract class is a class that can't be instantiated. That means you cannot create new instances of an abstract class. It works as a base for subclasses. You should learn about Java Inheritance before attempting this challenge.

Following is an example of abstract class:

```
abstract class Book{
    String title;
    abstract void setTitle(String s);
    String getTitle(){
        return title;
    }
}
```

If you try to create an instance of this class like the following line you will get an error:

```
Book new_novel=new Book();
```

You have to create another class that extends the abstract class. Then you can create an instance of the new class.

Notice that setTitle method is abstract too and has no body. That means you must implement the body of that method in the child class.

In the editor, we have provided the abstract Book class and a Main class. In the Main class, we created an instance of a class called MyBook. Your task is to write just the MyBook class.

Your class mustn't be public.

Sample Input

```
A tale of two cities
```

Sample Output

```
The title is: A tale of two cities
```

Submitted Code

Language: Java 7 — Open in editor

```
9
10
11 //Write MyBook class here
12 class MyBook extends Book {
13     void setTitle(String s) {
14         this.title = s;
15     }
16 }
17
18
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
1 A tale of two cities
```

Expected Output — Download

```
1 The title is: A tale of two cities
```

Q44. Java Interface

A Java interface can only contain method signatures and fields. The interface can be used to achieve polymorphism. In this problem, you will practice your knowledge on interfaces.

You are given an interface AdvancedArithmetic which contains a method signature int divisor_sum(int n). You need to write a class called MyCalculator which implements the interface.

divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.

Read the partially completed code in the editor and complete it. You just need to write the MyCalculator class only. Your class shouldn't be public.

Sample Input

```
6
```

Sample Output

```
I implemented: AdvancedArithmetic
12
```

Explanation

Divisors of 6 are 1,2,3 and 6. 1+2+3+6=12.

Submitted Code

Language: Java 7 — Open in editor

```
5
6
7  class MyCalculator implements AdvancedArithmetic {
8      public int divisor_sum(int n) {
9          int sum = 0;
10         for (int i = 1; i <= n; i++) {
11             if (n % i == 0) {
12                 sum += i;
13             }
14         }
15         return sum;
16     }
17 }
18
19
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin) — Download

```
1  6
```

Expected Output — Download

```
1  I implemented: AdvancedArithmetic
2  12
```

Q45. Java Method Overriding

When a subclass inherits from a superclass, it also inherits its methods; however, it can also override the superclass methods (as well as declare and implement new ones). Consider the following Sports class:

```
class Sports{
    String getName(){
        return "Generic Sports";
    }
    void getNumberOfTeamMembers(){
        System.out.println( "Each team has n players in " + getName() );
    }
}
```

Next, we create a Soccer class that inherits from the Sports class. We can override the getName method and return a different, subclass-specific string:

```
class Soccer extends Sports{
    @Override
    String getName(){
        return "Soccer Class";
    }
}
```

Note: When overriding a method, you should precede it with the `@Override` annotation. The parameter(s) and return type of an overridden method must be exactly the same as those of the method inherited from the supertype.

Task

Complete the code in your editor by writing an overridden getNumberOfTeamMembers method that prints the same statement as the superclass' getNumberOfTeamMembers method, except that it replaces n with 11 (the number of players on a Soccer team).

Output Format

When executed, your completed code should print the following:

Submitted Code

Language: Java 7 — Open in editor

```
20
21
22  @Override
23  void getNumberOfTeamMembers(){
24      System.out.println("Each team has 11 players in " + getName());
25  }
26
27
```

Test case 0

Compiler Message

Success

Expected Output — Download

```
1  Generic Sports
2  Each team has n players in Generic Sports
3  Soccer Class
4  Each team has 11 players in Soccer Class
```

Q46. Java Method Overriding 2 (Super Keyword)

When a method in a subclass overrides a method in superclass, it is still possible to call the overridden method using **super** keyword. If you write super.func() to call the function func(), it will call the method that was defined in the superclass.

You are given a partially completed code in the editor. Modify the code so that the code prints the following text:

```
Hello I am a motorcycle, I am a cycle with an engine.
My ancestor is a cycle who is a vehicle with pedals.
```

Submitted Code

Language: Java 7 — Open in editor

```
18
19
20 String temp = super.define_me();
21
22
```

Test case 0

Compiler Message

```
Success
```

Expected Output — Download

```
1 Hello I am a motorcycle, I am a cycle with an engine.
2 My ancestor is a cycle who is a vehicle with pedals.
```

Q47. Java Instanceof keyword

The Java instanceof operator is used to test if the object or instance is an instanceof the specified type.

In this problem we have given you three classes in the editor:

- Student class
- Rockstar class
- Hacker class

In the main method, we populated an ArrayList with several instances of these classes. count method calculates how many instances of each type is present in the ArrayList. The code prints three integers, the number of instance of Student class, the number of instance of Rockstar class, the number of instance of Hacker class.

But some lines of the code are missing, and you have to fix it by modifying only **3** lines! Don't add, delete or modify any extra line.

To restore the original code in the editor, click on the top left icon in the editor and create a new buffer.

Sample Input

```
5
Student
Student
Rockstar
Student
Hacker
```

Sample Output

```
3 1 1
```

Submitted Code

Language: Java 7 — Open in edito

```
26
27  public static void main(String []args){
28     ArrayList mylist = new ArrayList();
29     Scanner sc = new Scanner(System.in);
30     int t = sc.nextInt();
31     for(int i=0; i<t; i++){
32        String s=sc.next();
33        if(s.equals("Student"))mylist.add(new Student());
34        if(s.equals("Rockstar"))mylist.add(new Rockstar());
35        if(s.equals("Hacker"))mylist.add(new Hacker());
36     }
37     System.out.println(count(mylist));
38  }
39 }
40
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
5
Student
Student
Rockstar
```

Q48. Java Iterator

Java Iterator class can help you to iterate through every element in a collection. Here is a simple example:

```
import java.util.*;
public class Example{

    public static void main(String []args){
        ArrayList mylist = new ArrayList();
        mylist.add("Hello");
        mylist.add("Java");
        mylist.add("4");
        Iterator it = mylist.iterator();
        while(it.hasNext()){
            Object element = it.next();
            System.out.println((String)element);
        }
    }
}
```

In this problem you need to complete a method func. The method takes an ArrayList as input. In that ArrayList there is one or more integer numbers, then there is a special string "###", after that there are one or more other strings. A sample ArrayList may look like this:

```
element[0]=>42
element[1]=>10
element[2]=>"###"
element[3]=>"Hello"
element[4]=>"Java"
```

You have to modify the func method by editing `at most 2 lines` so that the code only prints the elements after the special string "###". For the sample above the output will be:

```
Hello
Java
```

Note: The stdin doesn't contain the string "###", it is added in the main method.

To restore the original code in the editor, click the top left icon on the editor and create a new buffer.

Submitted Code

Language: Java 7 — Open in edit

```
7
8       Object element = it.next(); // Line 1 fixed
9      if(element instanceof String && element.equals("###")) // Line 2 fixed
10
```

Test case 0

Test case 1

Test case 2

Compiler Message

```
Success
```

Input (stdin) — Download

```
1  2 2
2  42
3  10
4  hello
5  iava
```

Q49. Java Exception Handling (Try-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
```

Sample Input 1:

```
10
Hello
```

Sample Output 1:

```
java.util.InputMismatchException
```

Submitted Code

Language: Java 7 — Open in editor

```
7       Scanner in = new Scanner(System.in);
8       try {
9           int first = in.nextInt();
10          int second = in.nextInt();
11          System.out.println(first/second);
12
13      } catch (InputMismatchException e){
14          System.out.println(e.getClass().getName());
15      } catch (Exception e) {
16          System.out.println(e);
17      }
18      in.close();
19  }
20 }
21
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

Success

Input (stdin) — Download

```
10
3
```

Expected Output — Download

```
3
```

Q50. Java Exception Handling

You are required to compute the power of a number by implementing a calculator. Create a class MyCalculator which consists of a single method `long power(int, int)`. This method takes two integers, n and p, as parameters and finds n^p. If either n or p is negative, then the method must throw an exception which says "**`n or p should not be negative`**". Also, if both n and p are zero, then the method must throw an exception which says "**`n and p should not be zero.`**"

For example, -4 and -5 would result in **`java.lang.Exception: n or p should not be negative`**.

Complete the function `power` in class MyCalculator and return the appropriate result after the power operation or an appropriate exception as detailed above.

Input Format

Each line of the input contains two integers, n and p. The locked stub code in the editor reads the input and sends the values to the method as parameters.

Constraints

- $-10 \leq n \leq 10$
- $-10 \leq p \leq 10$

Output Format

Each line of the output contains the result n^p, if both n and p are positive. If either n or p is negative, the output contains "n and p should be non-negative". If both n and p are zero, the output contains "n and p should not be zero.". This is printed by the locked stub code in the editor.

Submitted Code

Language: Java 7 — Open in edi

```
3
4  class MyCalculator {
5  long power(int first, int second) throws Exception {
6        if(first < 0 || second < 0){
7           throw new Exception("n or p should not be negative.");
8        }
9        else if (first == 0 && second == 0 ){
10          throw new Exception("n and p should not be zero.");
11       }
12       else{
13          return (long)Math.pow(first,second);
14       }
15    }
16
17 }
```

Test case 0

Test case 1

Compiler Message

Success

Input (stdin) — Download

```
3 5
2 4
0 0
-1 -2
-1 3
```

Q51. Java Varargs - Simple Addition

You are given a class Solution and its main method in the editor.

Your task is to create the class Add and the required methods so that the code prints the sum of the numbers passed to the function add.

Note: Your add method in the Add class must print the sum as given in the Sample Output

Input Format

There are six lines of input, each containing an integer.

Output Format

There will be only four lines of output. Each line contains the sum of the integers passed as the parameters to add in the main method.

Sample Input

```
1
2
3
4
5
6
```

Sample Output

```
1+2=3
1+2+3=6
1+2+3+4+5=15
1+2+3+4+5+6=21
```

Submitted Code

Language: Java 7

```
class Add {
    public void add(int... intArgs) {
        int sum = 0;
        String separator = "";
        for (int i : intArgs) {
            sum += i;
            System.out.print(separator + i);
            separator = "+";
        }
        System.out.println("=" + sum);
    }
}
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Input (stdin)

```
1
2
3
4
5
6
```

Q52. Java Reflection - Attributes

JAVA reflection is a very powerful tool to inspect the attributes of a class in runtime. For example, we can retrieve the list of public fields of a class using getDeclaredMethods().

In this problem, you will be given a class Solution in the editor. You have to fill in the incompleted lines so that it prints all the methods of another class called Student in alphabetical order. We will append your code with the Student class before running it.

The Student class looks like this:

```
class Student{
    private String name;
    private String id;
    private String email;

    public String getName() {
        return name;
    }
    public void setId(String id) {
        this.id = id;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void anothermethod(){  }
    ......
    ......
    some more methods
    ......
}
```

You have to print all the methods of the student class in alphabetical order like this:

```
anothermethod
getName
setEmail
setId
......
......
some more methods
......
```

There is no sample input/output for this problem. If you press "Run Code", it will compile it, but it won't show any outputs.

Submitted Code

Language: Java 7

```
88      Class student = Student.class;
89
90      // Get all declared methods (including private methods)
91      Method[] methods = student.getDeclaredMethods();
92
93      // Create an ArrayList to store method names
94      ArrayList<String> methodList = new ArrayList<>();
95
96      // Loop through the methods and add their names to the list
97      for(Method method : methods){
98          methodList.add(method.getName());
99      }
100
101     // Sort the method names in alphabetical order
102     Collections.sort(methodList);
```

Test case 0

Compiler Message

Success

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Q53. Can You Access?

You are given a class Solution and an inner class Inner.Private. The main method of class Solution takes an integer num as input. The powerof2 in class Inner.Private checks whether a number is a power of 2. You have to call the method powerof2 of the class Inner.Private from the main method of the class Solution.

Constraints

$1 \leq num \leq 2^{30}$

Sample Input

```
8
```

Sample Output

```
8 is power of 2
An instance of class: Solution.Inner.Private has been created
```

Submitted Code

Language: Java 7 — Open in editor

```
17
18
19      Solution.Inner.Private innerPrivate = new Solution.Inner().new Private();
20
21      // Assigning the instance to Object o
22      o = innerPrivate;
23
24      // Calling powerof2 method and printing the result
25      System.out.println(num + " is " + innerPrivate.powerof2(num));
26
27
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin) — Download

```
7
```

Expected Output — Download

```
7 is not a power of 2
An instance of class: Solution.Inner.Private has been created
```

Q54. Prime Checker

You are given a class Solution and its main method in the editor. Your task is to create a class Prime. The class Prime should contain a single method checkPrime.

The locked code in the editor will call the checkPrime method with one or more integer arguments. You should write the checkPrime method in such a way that the code prints only the prime numbers.

Please read the code given in the editor carefully. Also please do not use method overloading!

Note: You may get a compile time error in this problem due to the statement below:

```
BufferedReader br=new BufferedReader(new InputStreamReader(in));
```

This was added intentionally, and you have to figure out a way to get rid of the error.

Input Format

There are only five lines of input, each containing one integer.

Output Format

There will be only four lines of output. Each line contains only prime numbers depending upon the parameters passed to checkPrime in the main method of the class Solution. In case there is no prime number, then a blank line should be printed.

Sample Input

```
2
1
3
4
5
```

Sample Output

```
2
2
2 3
2 3 5
```

Submitted Code

Language: Java 7

```
// Fix for BufferedReader issue by defining 'in' properly
import static java.lang.System.in;

// Define the Prime class
class Prime {
    void checkPrime(int... numbers) { // Using varargs to avoid overloading
        boolean hasPrime = false;
        for (int num : numbers) {
            if (isPrime(num)) {
                System.out.print(num + " ");
                hasPrime = true;
            }
        }
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Compiler Message

Success

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Q55. Java Factory Pattern

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
```

Sample Output 2

```
The factory returned class Pizza
Someone ordered Fast Food!
```

Submitted Code

Language: Java 7 — Open in editor

```
22
23
24 if (order.equalsIgnoreCase("pizza")) {
25             return new Pizza();
26         } else if (order.equalsIgnoreCase("cake")) {
27             return new Cake();
28         }
29         return null;
30
31
32
```

Test case 0

Test case 1

Compiler Message

Success

Input (stdin) — Download

```
1 cake
```

Expected Output — Download

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

Q56. Java Singleton Pattern

> "The singleton pattern is a design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system."
> - Wikipedia: Singleton Pattern

Complete the Singleton class in your editor which contains the following components:

1. A private Singleton non parameterized constructor.
2. A public String instance variable named str.
3. Write a static method named getSingleInstance that returns the single instance of the Singleton class.

Once submitted, our hidden Solution class will check your code by taking a String as input and then using your Singleton class to print a line.

Input Format

You will not be handling any input in this challenge.

Output Format

You will not be producing any output in this challenge.

Sample Input

```
hello world
```

Sample Output

```
Hello I am a singleton! Let me say hello world to you
```

Submitted Code

Language: Java 7

```
3  import java.io.*;
4  import java.util.*;
5  import java.text.*;
6  import java.math.*;
7  import java.util.regex.*;
8  import java.lang.reflect.*;
9
10
11 class Singleton{
12 // Step 1: Create a private static instance of Singleton
13    private static Singleton singleInstance = new Singleton();
14
15    // Step 2: Public string variable
16    public String str;
17
```

Test case 0

Test case 1

Compiler Message

Success

Input (stdin)

```
1  hello world
```

Q57. Java Visitor Pattern

Part II: Read and Build the Tree

Read the n-node tree, where each node is numbered from 1 to n. The tree is given as a list of node values $(x_1, x_2, \ldots, x_n)$, a list of node colors $(c_1, c_2, \ldots, c_n)$, and a list of edges. Construct this tree as an instance of the Tree class. The tree is always rooted at node number 1.

Your implementations of the three visitor classes will be tested on the tree you built from the given input.

Input Format

The first line contains a single integer, n, denoting the number of nodes in the tree. The second line contains n space-separated integers describing the respective values of $x_1, x_2, \ldots, x_n$.

The third line contains n space-separated binary integers describing the respective values of $c_1, c_2, \ldots, c_n$. Each c_i denotes the color of the i^{th} node, where 0 denotes red and 1 denotes green.

Each of the $n-1$ subsequent lines contains two space-separated integers, u_i and v_i, describing an edge between nodes u_i and v_i.

Constraints

- $2 \leq n \leq 10^5$
- $1 \leq x_i \leq 10^3$
- $c_i \in \{0, 1\}$
- $1 \leq v_i, u_i \leq n$
- It is guaranteed that the tree is rooted at node 1.

Output Format

Do not print anything to stdout, as this is handled by locked stub code in the editor. The three `getResult()` methods provided for you must return an integer denoting the $result$ for that class' visitor (defined above). Note that the value returned by ProductRedNodesVisitor's getResult method must be computed modulo $10^9 + 7$.

Submitted Code

Language: Java 7

```
141         sumOfValuesGreenLeaf += leaf.getValue();
142     }
143 }
144
145 public class JavaVisitorPattern {
146         //read the tree from STDIN and return its root as a return value of this function
147     static Map<Integer, Tree> tree = new HashMap<>();
148     public static Tree solve() {
149         Scanner sc = new Scanner(System.in);
150         int n = sc.nextInt();
151         Map<Integer, Object[]> nodeAtts = new HashMap<Integer, Object[]>();
152
153         for (int i = 0; i < n; i++)
154             nodeAtts.put(i + 1, new Object[]{sc.nextInt(), null});
155
156         for (int i = 0; i < n; i++)
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin)

```
5
4 7 2 5 12
0 1 0 0 1
1 2
1 3
3 4
3 5
```

Expected Output

Q58. Java Annotations

Here, we partially define an annotation, **FamilyBudget** and a class, **FamilyMember**. In this problem, we give the user role and the amount of money that a user spends as inputs. Based on the user role, you have to call the appropriate method in the **FamilyMember** class. If the amount of money spent is over the budget limit for that user role, it prints `Budget Limit Over`.

Your task is to complete the **FamilyBudget** annotation and the **FamilyMember** class so that the **Solution** class works perfectly with the defined constraints.

Note: You must complete the **5** incomplete lines in the editor. You are not allowed to change, delete or modify any other lines. To restore the original code, click on the top-left button on the editor and create a new buffer.

Input Format

The first line of input contains an integer N representing the total number of test cases. Each test case contains a string and an integer separated by a space on a single line in the following format:

```
UserRole MoneySpend
```

Constraints

$2 \leq N \leq 10$

$0 \leq MoneySpend \leq 200$

$|UserRole| = 6$

Name contains only lowercase English letters.

Output Format

Based on the user role and budget outputs, output the contents of the certain method. If the amount of money spent is over the budget limit, then output `Budget Limit Over`.

Submitted Code

Language: Java 7

```
33      String role = in.next();
34      int spend = in.nextInt();
35      try {
36        Class annotatedClass = FamilyMember.class;
37        Method[] methods = annotatedClass.getMethods();
38        for (Method method : methods) {
39          if (method.isAnnotationPresent(FamilyBudget.class)) {
40            FamilyBudget family = method
41                .getAnnotation(FamilyBudget.class);
42            String userRole = family.userRole();
43            int budgetLimit = family.budgetLimit();
44            if (userRole.equals(role)) {
45              if(spend<=budgetLimit){
46                method.invoke(FamilyMember.class.newInstance(),
47                    budgetLimit, spend);
```

Test case 0

Test case 1

Test case 2

Test case 3

Compiler Message

```
Success
```

Input (stdin)

```
3
SENIOR 75
JUNIOR 45
SENIOR 40
```

Expected Output

```
Senior Member
Spend: 75
Budget Left: 25
```

Q59. Covariant Return Types

You will be given a partially completed code in the editor where the main method takes the name of a state (i.e., `WestBengal`, or `AndhraPradesh`) and prints the national flower of that state using the classes and methods written by you.

Note: Do not use access modifiers in your class declarations.

Resources

Covariant Return Type

Java Covariant Type

Input Format

The locked code reads a single string denoting the name of a subclass of State (i.e., `WestBengal`, `Karnataka`, or `AndhraPradesh`), then tests the methods associated with that subclass. You are not responsible for reading any input from stdin.

Output Format

Output is handled for you by the locked code, which creates the object corresponding to the input string's class name and then prints the name returned by that class' national flower's whatsYourName method. You are not responsible for printing anything to stdout.

Sample Input 0

```
AndhraPradesh
```

Sample Output 0

```
Lily
```

Explanation 0

An AndhraPradesh object's yourNationalFlower method returns an instance of the Lily class, and the Lily class' whatsYourName method returns `Lily`, which is printed by the hidden code checker.

Submitted Code

Language: Java 7 — Open in editor

```
5
6
7  class Flower {
8      String whatsYourName() {
9          return "I have many names and types.";
10     }
11 }
12
13 class Jasmine extends Flower {
14     @Override
15     String whatsYourName() {
16         return "Jasmine";
17     }
18 }
19
```

Test case 0

Test case 1

Compiler Message

Success

Input (stdin) — Download

```
1  AndhraPradesh
```

Expected Output — Download

```
1  Lily
```

Q60. Java Lambda Expressions

This Java 8 challenge tests your knowledge of Lambda expressions!

Write the following methods that return a lambda expression performing a specified action:

1. PerformOperation isOdd(): The lambda expression must return $true$ if a number is odd or $false$ if it is even.
2. PerformOperation isPrime(): The lambda expression must return $true$ if a number is prime or $false$ if it is composite.
3. PerformOperation isPalindrome(): The lambda expression must return $true$ if a number is a palindrome or $false$ if it is not.

Input Format

Input is handled for you by the locked stub code in your editor.

Output Format

The locked stub code in your editor will print T lines of output.

Sample Input

The first line contains an integer, T (the number of test cases).

The T subsequent lines each describe a test case in the form of 2 space-separated integers:

The first integer specifies the condition to check for (1 for Odd/Even, 2 for Prime, or 3 for Palindrome). The second integer denotes the number to be checked.

```
5
1 4
2 5
3 898
1 3
2 12
```

Sample Output

```
EVEN
PRIME
PALINDROME
ODD
COMPOSITE
```

Submitted Code

Language: Java 7 — Open in editor

```
49
50      while (T-- > 0) {
51          String[] s = br.readLine().split(" ");
52          int ch = Integer.parseInt(s[0]);
53          int num = Integer.parseInt(s[1]);
54          PerformOperation op;
55          boolean ret = false;
56          String ans = null;
57
58          switch (ch) {
59              case 1:
60                  op = ob.isOdd();
61                  ret = ob.checker(op, num);
62                  ans = (ret) ? "ODD" : "EVEN";
63                  break;
64              case 2:
```

Test case 0

Test case 1

Test case 2

Compiler Message

Success

Input (stdin) — Download

```
1 5
2 1 4
3 2 5
4 3 898
5 1 3
6 2 12
```

Q61. Java MD5

MD5 (Message-Digest algorithm 5) is a widely-used cryptographic hash function with a 128-bit hash value. Here are some common uses for MD5:

- To store a one-way hash of a password.
- To provide some assurance that a transferred file has arrived intact.

MD5 is one in a series of message digest algorithms designed by Professor Ronald Rivest of MIT (Rivest, 1994); however, the security of MD5 has been severely compromised, most infamously by the Flame malware in 2012. The CMU Software Engineering Institute essentially considers MD5 to be "cryptographically broken and unsuitable for further use".

Given an alphanumeric string, s, denoting a password, compute and print its MD5 encryption value.

Input Format

A single alphanumeric string denoting s.

Constraints

- $6 \leq |s| \leq 20$
- String s consists of English alphabetic letters (i.e., $[a - zA - Z]$ and/or decimal digits (i.e., 0 through 9) only.

Output Format

Print the MD5 encryption value of s on a new line.

Sample Input 0

```
HelloWorld
```

Sample Output 0

```
68e109f0f40ca72a15e05cc22786f8e6
```

Sample Input 1

```
Javarmi123
```

Sample Output 1

```
2da2d1e0ce7b4951a858ed2d547ef485
```

Submitted Code

Language: Java 7

```
12      sc.close();
13      try {
14          MessageDigest md = MessageDigest.getInstance("MD5");
15          md.update(str.getBytes());
16          // return bytesToHex(md.digest
17          byte[] digest = md.digest();
18          for (byte b : digest) {
19              System.out.printf("%02x", b);
20          }
21      } catch (Exception ex) {
22          throw new RuntimeException(ex);
23      }
24  }
25 }
26
```

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

```
Success
```

Input (stdin)

```
1  HelloWorld
```

Expected Output

```
1  68e109f0f40ca72a15e05cc22786f8e6
```

Q62. Java SHA-256

Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed hash (i.e., the ou produced by executing a hashing algorithm) to a known and expected hash value, a person can determine the data's integrit For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can sh whether the download has been modified or tampered with. In addition, cryptographic hash functions are extremely collisior resistant; in other words, it should be extremely difficult to produce the same hash output from two different input values usi cryptographic hash function.

Secure Hash Algorithm 2 (SHA-2) is a set of cryptographic hash functions designed by the National Security Agency (NSA). It consists of six identical hashing algorithms (i.e., SHA-256, SHA-512, SHA-224, SHA-384, SHA-512/224, SHA-512/256) with a variable digest size. SHA-256 is a 256-bit (32 byte) hashing algorithm which can calculate a hash code for an input of up to $264 - 1$ bits. It undergoes 64 rounds of hashing and calculates a hash code that is a 64-digit hexadecimal number.

Given a string, s, print its SHA-256 hash value.

Input Format

A single alphanumeric string denoting s.

Constraints

- $6 \leq |s| \leq 20$
- String s consists of English alphabetic letters (i.e., $[a - zA - Z]$ and/or decimal digits (i.e., 0 through 9) only.

Output Format

Print the SHA-256 encryption value of s on a new line.

Sample Input 0

```
HelloWorld
```

Sample Output 0

```
872e4e50ce9990d8b041330c47c9ddd11bec6b503ae9386a99da8584e9bb12c4
```

Sample Input 1

```
Javarmi123
```

Submitted Code

Language: Java 7 — Open in editor

```
7   public static void main(String[] args) throws NoSuchAlgorithmException {
8       /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. */
9
10      Scanner input = new Scanner(System.in);
11      MessageDigest m = MessageDigest.getInstance("SHA-256");
12      m.reset();
13      m.update(input.nextLine().getBytes());
14      input.close();
15      for (byte i : m.digest()) {
16          System.out.print(String.format("%02x", i));
17      }
18      System.out.println();
19  }
20 }
21
```

Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6

Compiler Message

```
Success
```

Input (stdin) — Download

```
1  HelloWorld
```

Expected Output — Download

```
1  872e4e50ce9990d8b041330c47c9ddd11bec6b503ae9386a99da8584e9bb12c4
```

Q63. Trim a Binary Search Tree

669. Trim a Binary Search Tree

Solved

Medium | Topics | Companies

Given the `root` of a binary search tree and the lowest and highest boundaries as `low` and `high`, trim the tree so that all its elements lies in `[low, high]`. Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a **unique answer**.

Return *the root of the trimmed binary search tree*. Note that the root may change depending on the given bounds.

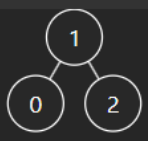
```java
class Solution {
    public TreeNode trimBST(TreeNode root, int low, int high) {

    }
}
```

Saved

Ln 1, Col

Testcase | Test Result



low =

1

high =

2

Q64. Add two Numbers II

445. Add Two Numbers II

Solved

Medium | Topics | Companies

You are given two **non-empty** linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

```cpp
class Solution {
public:
    ListNode* reverseLinkedList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* curr = head;
        while (curr) {
            ListNode* nextNode = curr->next;
            curr->next = prev;
            prev = curr;
            curr = nextNode;
        }
        return prev;
    }

    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        // Reverse both linked lists
        l1 = reverseLinkedList(l1);
        l2 = reverseLinkedList(l2);

        int carry = 0;
        ListNode* resultHead = new ListNode();
        ListNode* current = resultHead;

        while (l1 || l2) {
            int val1 = l1 ? l1->val : 0;
            int val2 = l2 ? l2->val : 0;

            int total = val1 + val2 + carry;
            carry = total / 10;

            current->next = new ListNode(total % 10);
```

Case 1 | Case 2 | Case 3 +

l1 =

[7,2,4,3]

l2 =

[5,6,4]

</> Source

Q65. Two Sum

1. Two Sum

Solved

Easy | Topics | Companies | Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Code | Accepted

All Submissions

Code | Java

```java
import java.util.HashMap;

class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> map = new HashMap<>();

        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];

            if (map.containsKey(complement)) {
                return new int[] { map.get(complement), i };
            }

            map.put(nums[i], i);
        }

        // According to the problem, one solution always exists
        return new int[] {};
    }
}
```

View less

Description | Editorial | Solutions | Su

Case 1 | Case 2 | Case 3 +

nums =

[2,7,11,15]

target =

9

Q66. Add two numbers

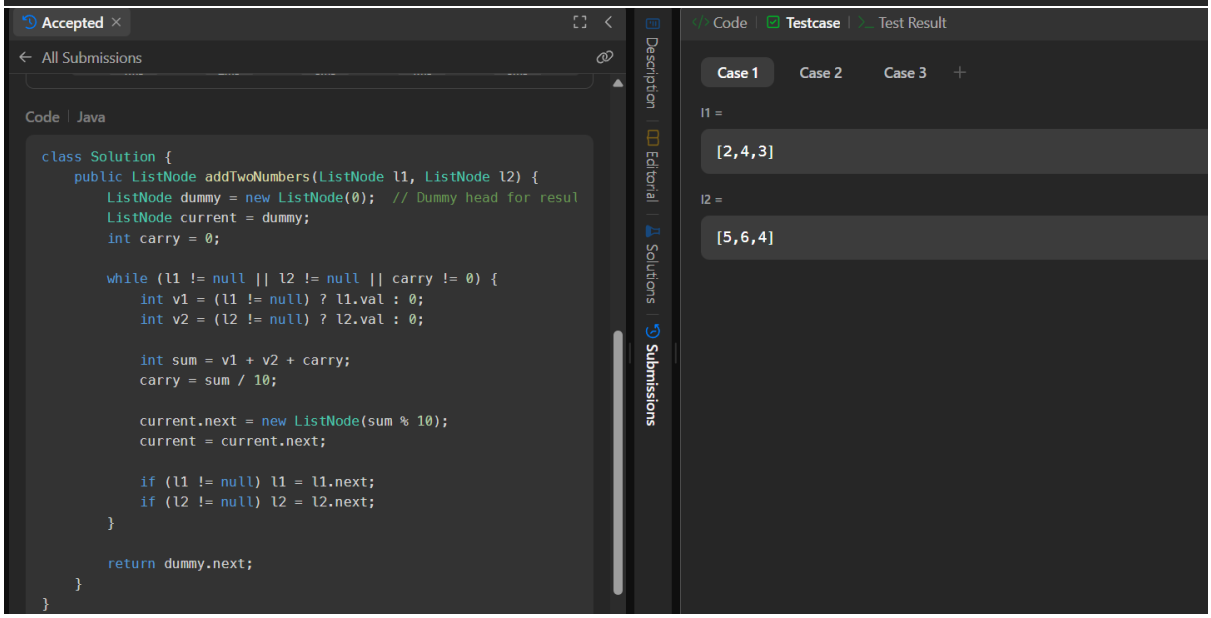
2. Add Two Numbers

Solved

Medium | Topics | Companies

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.



Q67. Count of Interesting Subarrays

2845. Count of Interesting Subarrays

Solved

Medium | Topics | Companies | Hint

You are given a **0-indexed** integer array `nums`, an integer `modulo`, and an integer `k`.

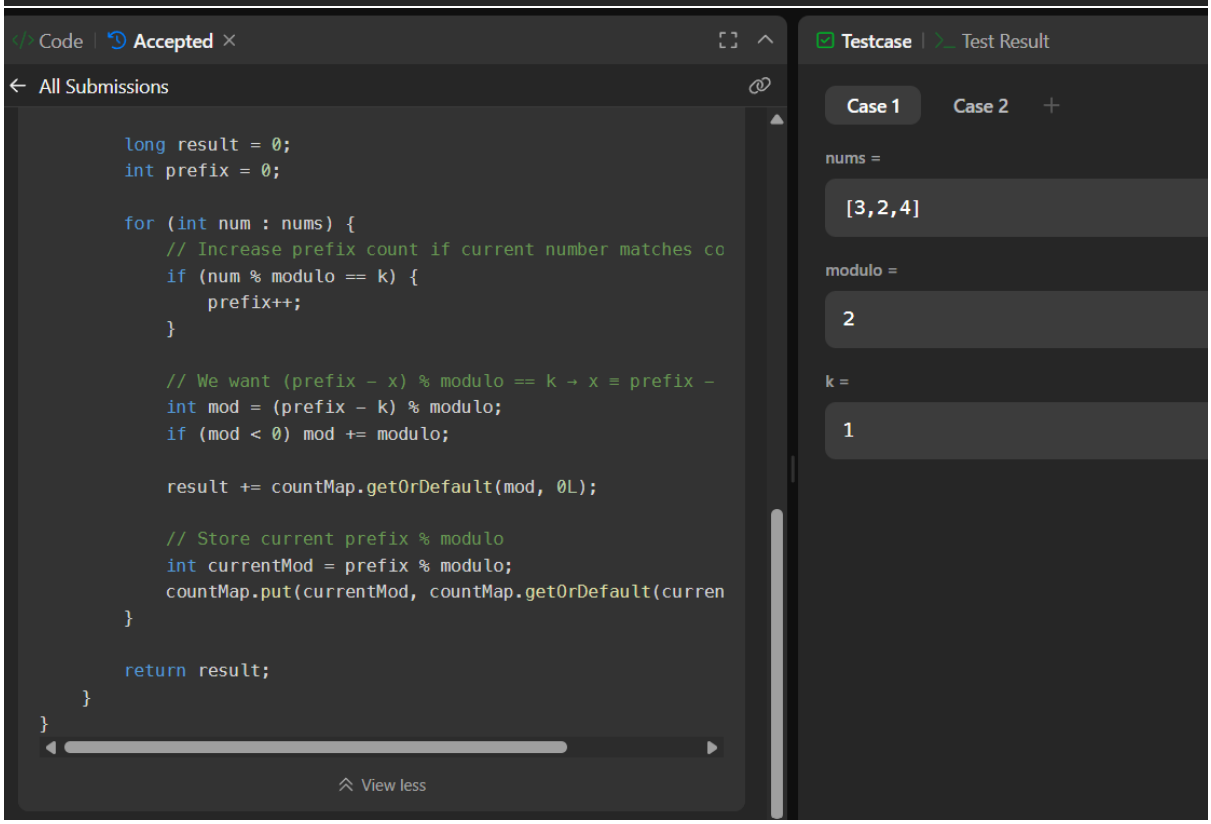
Your task is to find the count of subarrays that are **interesting**.

A **subarray** `nums[l..r]` is **interesting** if the following condition holds:

- Let `cnt` be the number of indices `i` in the range `[l, r]` such that `nums[i] % modulo == k`. Then, `cnt % modulo == k`.

Return *an integer denoting the count of interesting subarrays.*

Note: A subarray is *a contiguous non-empty sequence of elements within an array.*



Q68. Longest Substring without Repeating Characters

3. Longest Substring Without Repeating Characters

Solved

Medium | Topics | Companies | Hint

Given a string `s`, find the length of the **longest substring** without duplicate characters.

Example 1:

> **Input:** s = "abcabcbb"
> **Output:** 3
> **Explanation:** The answer is "abc", with the length of 3.

Example 2:

> **Input:** s = "bbbbb"
> **Output:** 1
> **Explanation:** The answer is "b", with the length of 1.

Example 3:

> **Input:** s = "pwwkew"
> **Output:** 3
> **Explanation:** The answer is "wke", with the length of 3.
> Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Code

Java | Auto

```
9           char current = s.charAt(right);
10
11          // If current char is already in the set, move left pointer until it's removed
12          while (seen.contains(current)) {
13              seen.remove(s.charAt(left));
14              left++;
15          }
16
17          seen.add(current);
18          maxLength = Math.max(maxLength, right - left + 1);
19      }
```

Saved Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | Case 3 +

s =

"abcabcbb"

Q69. Longest Palindromic Substring

5. Longest Palindromic Substring

Medium | Topics | Companies | Hint

Given a string `s`, return *the longest palindromic substring* in `s`.

Example 1:

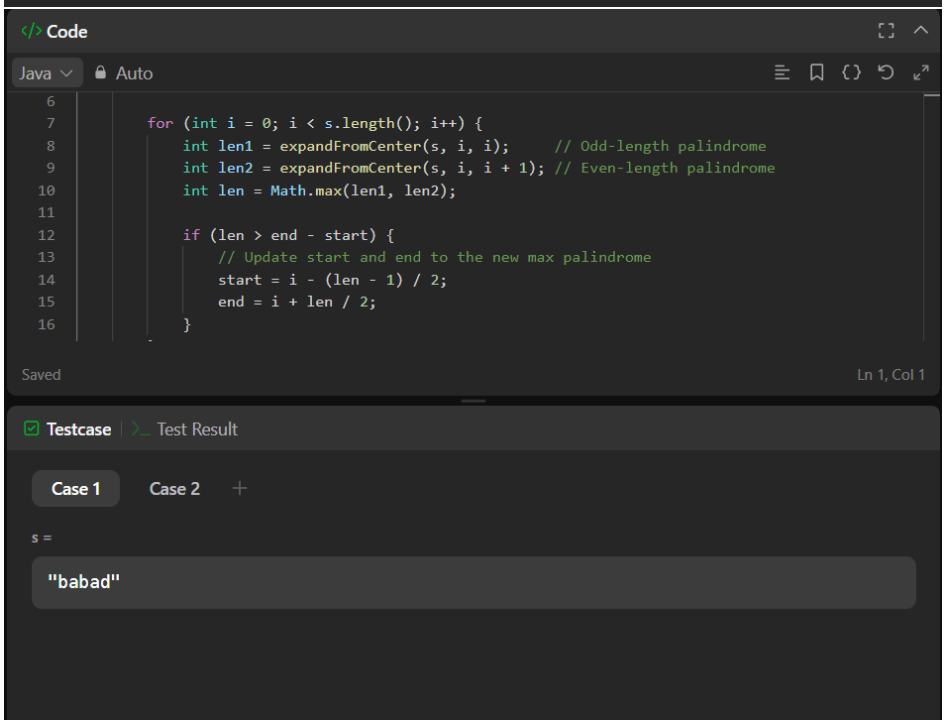
```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

Constraints:

- `1 <= s.length <= 1000`
- `s` consist of only digits and English letters.



Q70. Zigzag Conversion

6. Zigzag Conversion

Solved

Medium | Topics | Companies

The string `"PAYPALISHIRING"` is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

```
P   A   H   N
A P L S I I G
Y   I   R
```

And then read line by line: `"PAHNAPLSIIGYIR"`

Write the code that will take a string and make this conversion given a number of rows:

```
string convert(string s, int numRows);
```

Example 1:

```
Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"
```

Example 2:

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
```

Code

Java | Auto

```java
        int row = 0;
        boolean goingDown = false;

        for (char c : s.toCharArray()) {
            rows[row].append(c);

            // Change direction at top and bottom
            if (row == 0 || row == numRows - 1) goingDown = !goingDown;

            row += goingDown ? 1 : -1;
        }
```

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | Case 3 | +

s =

"PAYPALISHIRING"

numRows =

3

Q71. Regular Expression Matching

10. Regular Expression Matching

Solved

Hard | Topics | Companies

Given an input string `s` and a pattern `p`, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Example 1:

```
Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
```

Example 2:

```
Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by
repeating 'a' once, it becomes "aa".
```

Code

Java | Auto

```java
class Solution {
    public boolean isMatch(String s, String p) {
        int m = s.length(), n = p.length();

        // dp[i][j] = whether s[0..i-1] matches p[0..j-1]
        boolean[][] dp = new boolean[m + 1][n + 1];

        dp[0][0] = true; // empty string matches empty pattern

        // Handle patterns like a*, a*b*, a*b*c* matching an empty string
        for (int i = 2; i <= n; i++) {
```

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | Case 3 | +

s =

"aa"

p =

"a"

Q72. Integer to Roman

12. Integer to Roman

Solved

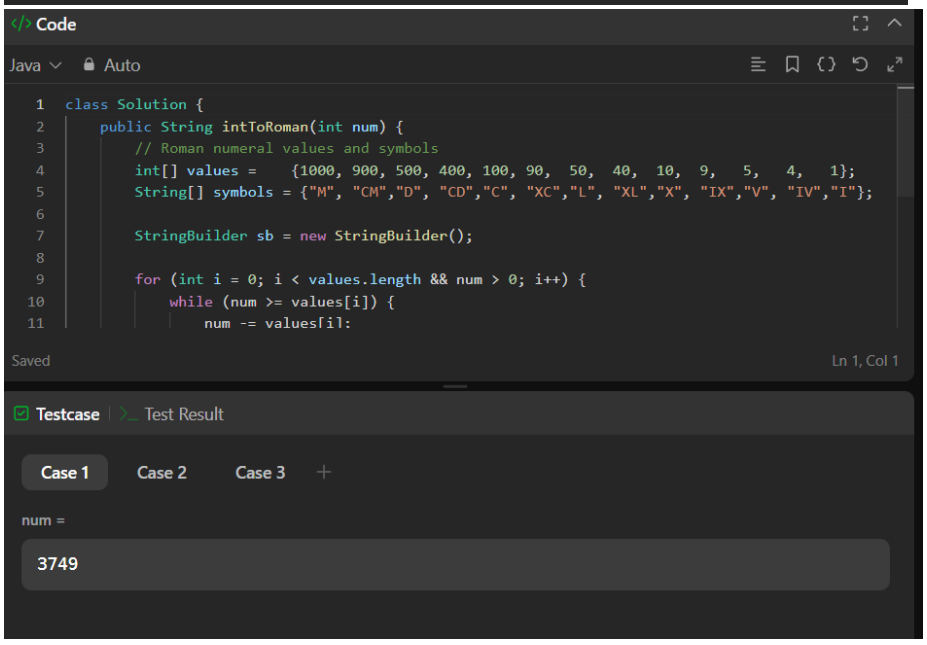
Medium | Topics | Companies

Seven different symbols represent Roman numerals with the following values:

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Roman numerals are formed by appending the conversions of decimal place values from highest to lowest. Converting a decimal place value into a Roman numeral has the following rules:

- If the value does not start with 4 or 9, select the symbol of the maximal value that can be subtracted from the input, append that symbol to the result, subtract its value, and convert the remainder to a Roman numeral.



Q73. Roman to Integer

13. Roman to Integer

Solved

Easy | Topics | Companies | Hint

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two ones added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

</> Code

Java | Auto

```java
class Solution {
    public int romanToInt(String s) {
        // Map Roman characters to values
        Map<Character, Integer> map = new HashMap<>();
        map.put('I', 1);  map.put('V', 5);
        map.put('X', 10); map.put('L', 50);
        map.put('C', 100); map.put('D', 500);
        map.put('M', 1000);

        int result = 0;
        int n = s.length();
```

Saved

Ln 1, Col 1

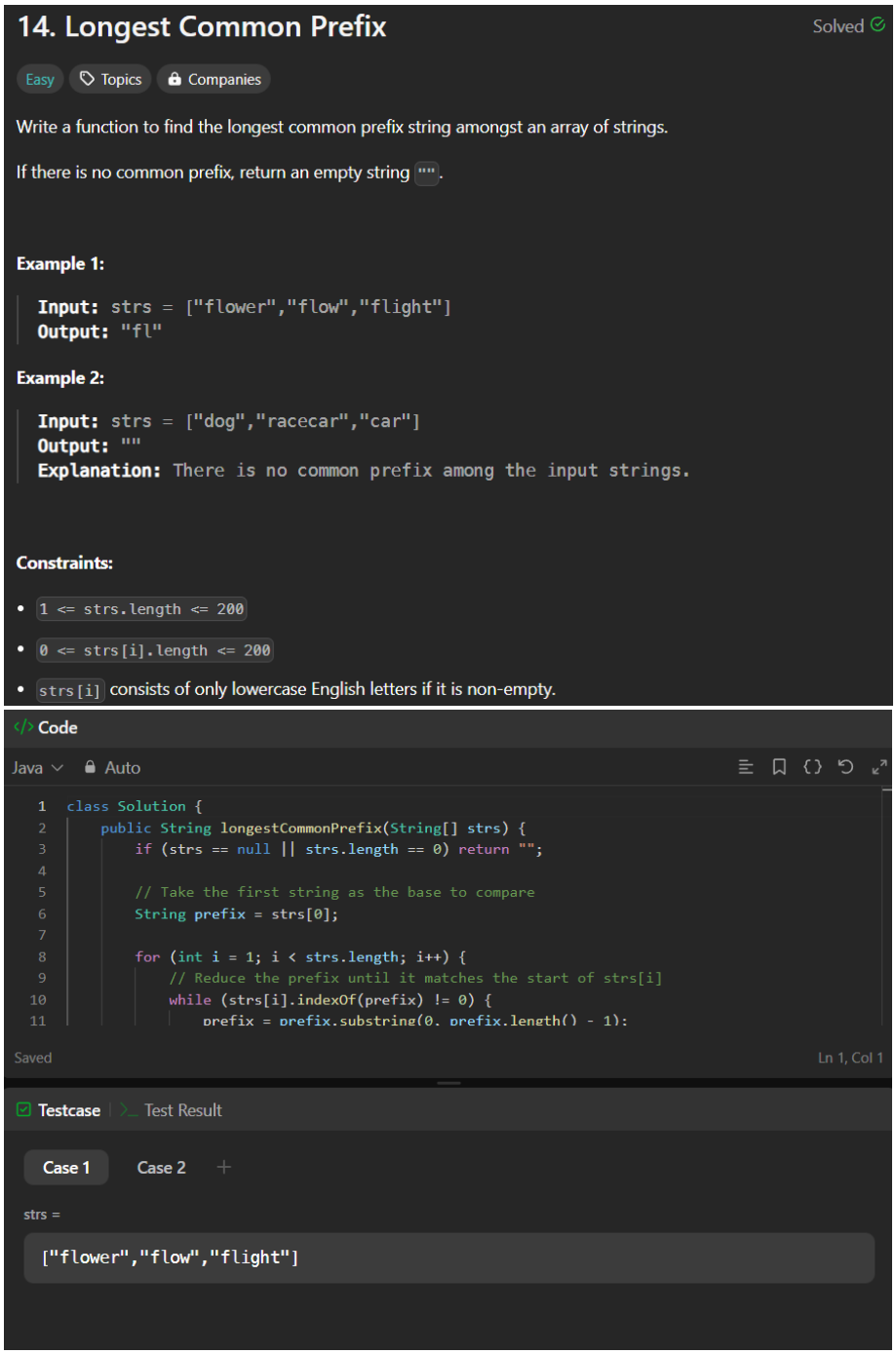
Testcase | Test Result

Case 1 | Case 2 | Case 3 | +

s =

"III"

Q74. Longest Common Prefix



14. Longest Common Prefix

Solved

Easy | Topics | Companies

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string `""`.

Example 1:

```
Input: strs = ["flower","flow","flight"]
Output: "fl"
```

Example 2:

```
Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.
```

Constraints:

- `1 <= strs.length <= 200`
- `0 <= strs[i].length <= 200`
- `strs[i]` consists of only lowercase English letters if it is non-empty.

Code

Java | Auto

```java
class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) return "";

        // Take the first string as the base to compare
        String prefix = strs[0];

        for (int i = 1; i < strs.length; i++) {
            // Reduce the prefix until it matches the start of strs[i]
            while (strs[i].indexOf(prefix) != 0) {
                prefix = prefix.substring(0, prefix.length() - 1);
```

Saved | Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | +

strs =

```
["flower","flow","flight"]
```

Q75. Median of two sorted arrays