1. **[Welcome to Java! – Hacker Rank Solution](https://www.codingbroz.com/welcome-to-java-solution-2/)**

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.

**Solution –** **Welcome to Java!**

**public** **class** Solution {

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

/\* Enter your code here. Print output to STDOUT. Your class should be named Solution. \*/

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

}

}

1. [**Java Stdin and Stdout I – Hacker Rank Solution**](https://www.codingbroz.com/java-stdin-and-stdout-1-solution/)

## ****Problem****

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

## ****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

## ****Solution –**** ****Java Stdin and Stdout 1****

**import** *java.util.*\*;

**public** **class** Solution {

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

Scanner scan = new Scanner(System.in);

**int** a = scan.nextInt();

**int** b = scan.nextInt();

**int** c = scan.nextInt();

System.out.println(a);

System.out.println(b);

System.out.println(c);

}

}

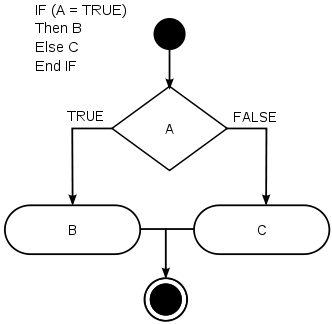
# Java If-Else | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-if-else-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java If-Else HackerRank Solution**.

## ****Problem****

In this challenge, we test your knowledge of using if-else conditional statements to automate decision-making processes. An if-else statement has the following logical flow:



## ****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

## ****Input Format****

A single line containing a positive integer, **n**.

## ****Constraints****

**1 <= n <= 100**

## ****Output Format****

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

**Sample input 0**

3

**Sample output 0**

Weird

**Sample input**1

24

**Sample output 1**

Not Weird

**Explanation**

Sample Case 0: ***n=3***, where ***n*** is odd and odd numbers are weird, so we print Weird.

## ****Solution –**** ****Java If-Else****

**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 {

**private** **static** **final** Scanner scanner = new Scanner(System.in);

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

**int** N = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

scanner.close();

**if** (N%2==1){

System.out.println("Weird");

}

**else** **if**(N%2==0 && (N>=2 && N<=5)){

System.out.println("Not Weird");

}

**else** **if**(N%2==0 && (N>5 && N<=20)){

System.out.println("Weird");

}

**else** **if**(N%2==0 && (N>20 && N<=100)){

System.out.println("Not Weird");

}

}

}

# Java Output Formatting | HackerRank Solution

[2 Comments](https://www.codingbroz.com/java-output-formatting-solution/#comments) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Output Formatting HackerRank Solution**.

## ****Problem****

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.

Conquering the clouds on a journey to Ta Xua with the team - Road Trip Vietnam Team - Nếm TV

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

## ****Input Format****

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

## ****Output Format****

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

**Sample Input**

java 100

cpp 65

python 50

**Sample Output**

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

java 100

cpp 065

python 050

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

**Explanation**

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

## ****Solution –**** ****Java Output Formatting****

**import** *java.util.Scanner*;

**public** **class** Solution {

**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();

System.out.printf("%-15s%03d\n", s1, x);

}

System.out.println("================================");

}

}

# Java Loops I | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-loops-1-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Loops I HackerRank Solution**.

## ****Problem****

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

Conquering the clouds on a journey to Ta Xua with the team - Road Trip Vietnam Team - Nếm TV00:14 / 02:5410 Sec

## ****Task****

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

## ****Input Format****

A single integer, **N**.

## ****Constraints****

**2 <=** **N <= 20**.

## ****Output Format****

Print **10**lines of output; each line **i**(where **1 <= i <= 10**) contains the **result**of **N X 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

## ****Solution –**** ****Java Loops 1****

**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 {

**private** **static** **final** Scanner scanner = new Scanner(System.in);

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

**int** N = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

**if**(N>1 && N<=20)

{

**for**(**int** i=1 ; i<=10 ; i++)

{

**int** mul=N\*i;

System.out.println(N+" x "+i+" = "+mul);

}

}

scanner.close();

}

}

# Java Loops II | HackerRank Solution

[1 Comment](https://www.codingbroz.com/java-loops-2-solution/#comments) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Loops II HackerRank Solution**.

## ****Problem****

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

Conquering the clouds on a journey to Ta Xua with the team - Road Trip Vietnam Team - Nếm TV

***( a+20 . b), ( a+20 . b +*21 . b*), . . . . . , ( a+2*0*. b + 21 . b + . . . + 2n-1 . 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 **ai**, **bi**, and **ni** values for that query.

## ****Constraints****

* **0 <= q <= 500**
* **0 <= a, b <= 50**
* **1 <= n <= 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

**Explanation**

We have two queries:

1. We use **a=0, b=2, and n=10** to produce some series **s0**, **s1**, **s2**,**. . . .**, **sn-1** :
   * **s0** = **0 + 1 . 2 = 2**
   * **s1** = **0 + 1 . 2 + 2 . 2 = 6**
   * **s2** = **0 + 1 . 2 + 2 . 2** **+ 4 . 2 = 14**
   * **…. and so on.**
   * Once we hit n = 10, we print the first 10 terms as a single line of space-separated integers.
2. We use **a=5, b=3, and n=5** to produce some series **s0**, **s1**, **s2**,**. . . .**, **sn-1** :
   * **s0** = **5 + 1 . 3 = 8**
   * **s1** =**5 + 1 . 3 + 2 . 3 = 14**
   * **s2** = **5 + 1 . 3 + 2 . 3** **+ 4 . 3 = 26**
   * **s2** = **5 + 1 . 3 + 2 . 3** **+ 4 . 3 + 8 . 3= 50**
   * **s2** = **5 + 1 . 3 + 2 . 3** ***+ 4 . 3*+ 8 . 3** **+ 16 . 3 = 98**

We print each element of our series as a single line of space-separated values.

## ****Solution –**** ****Java Loops 2****

import java.util.\*;

import java.io.\*;

class Solution{

public static void main(String []argh){

Scanner in = new Scanner(System.in);

int sum = 0;

int t=in.nextInt();

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

int a = in.nextInt();

int b = in.nextInt();

int n = in.nextInt();

if( (n>=1&&n<=15) && (a>=0&&a<=50) && (b>=0&&b<=50) ){

int j=0;

while(j<n){

int s = (int)Math.pow(2,j)\*b;

sum = sum+s;

int ss = sum+a;

System.out.print(ss + " ");

j++;

}

sum = 0;

int ss = 0;

}

System.out.println("");

}

in.close();

}

}

# Java Datatypes | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-datatypes-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Datatypes HackerRank Solution**.

## ****Problem****

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

Conquering the clouds on a journey to Ta Xua with the team - Road Trip Vietnam Team - Nếm TV

* 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.

## ****Input Format****

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

## ****Output Format****

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

n can be fitted in:

\* dataType

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

**byte < short < int < long ).**

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

n can't be fitted anywhere.

**Sample Input**

5

-150

150000

1500000000

213333333333333333333333333333333333

-100000000000000

**Sample Input**

-150 can be fitted in:

\* short

\* int

\* long

150000 can be fitted in:

\* int

\* long

1500000000 can be fitted in:

\* int

\* long

213333333333333333333333333333333333 can't be fitted anywhere.

-100000000000000 can be fitted in:

\* long

**Explanation**

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

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

## ****Solution –**** ****Java Datatypes****

**import** *java.util.*\*;

**import** *java.io.*\*;

**class** Solution{

**public** **static** **void** main(**String** []argh)

{

Scanner sc = new Scanner(System.in);

**int** t=sc.nextInt();

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

{

**try**

{

**long** x=sc.nextLong();

System.out.println(x+" can be fitted in:");

**if**(x>=-128 && x<=127)System.out.println("\* byte");

**if** (x>=-32768 && x<=32767)System.out.println("\* short");

**if** (x>=-2147483648 && x<=2147483647)System.out.println("\* int");

**if** (x>=-9223372036854775808L && x<=9223372036854775807L)System.out.println("\* long");

}

**catch**(Exception e)

{

System.out.println(sc.next()+" 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");

if (x>=-32768 && x<=32767)System.out.println("\* short");

if (x>=-2147483648 && x<=2147483647)System.out.println("\* int");

if (x>=-9223372036854775808L && x<=9223372036854775807L)System.out.println("\* long");

}

catch(Exception e)

{

System.out.println(sc.next()+" can't be fitted anywhere.");

}

}

}

}

# Java End-of-file | HackerRank Solution

Hello coders, today we are going to solve **Java End-of-file HackerRank Solution**.

## ****Problem****

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

Synthesize Beautiful Scenes of Ha Giang Via Super Quality Travel Video - Flycam Nem TV00:00 / 03:1710 Sec

**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.

## ****Solution –**** ****Java End-of-file****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner sc = new Scanner(System.in);

int c =1;

while(sc.hasNext()){

String s = sc.nextLine();

System.out.println(c+" "+s);

c++;

}

}

}

# Java Static Initializer Block | HackerRank Solution

Hello coders, today we are going to solve **Java Static Initializer Block HackerRank Solution**.

## ****Problem****

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

Synthesize Beautiful Scenes of Ha Giang Via Super Quality Travel Video - Flycam Nem TV00:03 / 03:1710 Sec

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 <= 0**or **H <= 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 <= B <= 100**
* **-100 <= H <= 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

**Sample Input 2**

-1

2

**Sample Output** **2**

java.lang.Exception: Breadth and height must be positive

## ****Solution –**** ****Java Static Initializer Block****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static int B,H;

private static boolean flag = false;

static

{

Scanner sc = new Scanner(System.in);

B = sc.nextInt();

H = sc.nextInt();

if(B>0 && H>0)

{

flag = true;

}

else

{

System.out.println("java.lang.Exception: Breadth and height must be positive");

}

}

public static void main(String[] args){

if(flag){

int area=B\*H;

System.out.print(area);

}

}//end of main

}//end of class

# Java Int to String | HackerRank Solution

Hello coders, today we are going to solve **Java Int to String HackerRank Solution**.

## ****Problem****

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

Exploring Ba Vi's Dragon Fruit Garden [NếmTV]

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**

100

**Sample Output**

Good job

## ****Solution –**** ****Java Int to String****

import java.util.\*;

import java.security.\*;

public class Solution {

public static void main(String[] args) {

DoNotTerminate.forbidExit();

try {

Scanner in = new Scanner(System.in);

int n = in .nextInt();

in.close();

//String s=???; Complete this line below

//Write your code here

String s = "" ;

if(n>=-100 && n<=100)

{

s = Integer.toString(n);

}

if (n == Integer.parseInt(s)) {

System.out.println("Good job");

} else {

System.out.println("Wrong answer.");

}

} catch (DoNotTerminate.ExitTrappedException e) {

System.out.println("Unsuccessful Termination!!");

}

}

}

//The following class will prevent you from terminating the code using exit(0)!

class DoNotTerminate {

public static class ExitTrappedException extends SecurityException {

private static final long serialVersionUID = 1;

}

public static void forbidExit() {

final SecurityManager securityManager = new SecurityManager() {

@Override

public void checkPermission(Permission permission) {

if (permission.getName().contains("exitVM")) {

throw new ExitTrappedException();

}

}

};

System.setSecurityManager(securityManager);

}

}

# Java Date and Time | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-date-and-time-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Date and Time HackerRank Solution**.

The [Calendar](https://www.codingbroz.com/java-date-and-time-solution/) class is an abstract class that provides methods for converting between a specific instant in time and a set of [calendar](https://www.codingbroz.com/java-date-and-time-solution/) fields such as YEAR, MONTH, DAY\_OF\_MONTH, HOUR, and so on, and for manipulating the calendar fields, such as getting the date of the next week.

Exploring Ba Vi's Dragon Fruit Garden [NếmTV]00:16 / 02:4810 Sec

Read about [***String to Date Conversion in Java***](https://www.codingbroz.com/java-string-to-date-conversion/) – [***here***](https://www.codingbroz.com/java-string-to-date-conversion/)

## ****Problem****

You are given a date. You just need to write the method, **getDay**, which returns the day on that date. To simplify your task, we have provided a portion of the code in the [editor](https://www.codingbroz.com/java-date-and-time-solution/).

**Example**

**month = 8** **day = 14 year = 2017**

The method should return **MONDAY**as the day on that date.



#### **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, **MM DD YYYY**respectively, in format.

### **Constraints**

**2000 <= year <= 3000**

**Sample Input**

08 05 2015

**Sample Output**

WEDNESDAY

#### **Explanation**

The day on August **15th** **2015**was WEDNESDAY.

## ****Solution –**** ****Java Date and Time****

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

import java.time.LocalDate;

//===========================SOLUTION==============================//

class Result {

/\*

\* Complete the 'findDay' function below.

\*

\* The function is expected to return a STRING.

\* The function accepts following parameters:

\* 1. INTEGER month

\* 2. INTEGER day

\* 3. INTEGER year

\*/

public static String findDay(int month, int day, int year) {

Calendar cal = Calendar.getInstance();

cal.set(Calendar.MONTH,month-1);

cal.set(Calendar.DAY\_OF\_MONTH,day);

cal.set(Calendar.YEAR,year);

String dayOfWeek = cal.getDisplayName(Calendar.DAY\_OF\_WEEK, Calendar.LONG, Locale.US).toUpperCase();

return dayOfWeek;

}

}

//=============================SOLUTION-END=============================//

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

int month = Integer.parseInt(firstMultipleInput[0]);

int day = Integer.parseInt(firstMultipleInput[1]);

int year = Integer.parseInt(firstMultipleInput[2]);

String res = Result.findDay(month, day, year);

bufferedWriter.write(res);

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

# Java Currency Formatter | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-currency-formatter-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Currency Formatter HackerRank Solution**.

## ****Problem****

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:

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

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 <= payment <= 109**

## ****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 **u**is **payment** formatted for Indian currency.  
On the third line, print China: c where **u**is **payment** formatted for Chinese currency.  
On the fourth line, print France: f, where **u**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.

## ****Solution –**** ****Java Currency Formatter****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

double payment = scanner.nextDouble();

scanner.close();

// Write your code here.

String us = NumberFormat.getCurrencyInstance(Locale.US).format(payment);

String india = NumberFormat.getCurrencyInstance(new Locale("en","in")).format(payment);

String china = NumberFormat.getCurrencyInstance(Locale.CHINA).format(payment);

String france = NumberFormat.getCurrencyInstance(Locale.FRANCE).format(payment);

System.out.println("US: " + us);

System.out.println("India: " + india);

System.out.println("China: " + china);

System.out.println("France: " + france);

}

}

# Java Strings Introduction | HackerRank Solution

Hello coders, today we are going to solve **Java Strings Introduction**[**HackerRank**](https://www.codingbroz.com/java-strings-introduction-solution/)**Solution**.

"A string is traditionally a sequence of characters, either as a literal constant or as some kind of variable." — Wikipedia: String (computer science)

## ****Problem****

This exercise is to test your understanding of Java Strings. A sample String declaration:

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

String myString = "Hello World!"

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 operation:

1. Sum of length of **A**and **B**.
2. Determine if **A**is lexicographically larger than **B**(i.e.: **B**comes after **A** in dictionary?).
3. Capitalize the first letter of **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 is **A**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**

hello

java

##### **Sample Input**

9

No

Hello Java

##### **Explanation**

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”.

## ****Solution –**** ****Java Strings Introduction****

import java.io.\*;

import java.util.\*;

public class Solution {

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. \*/

int len\_A = A.length();

int len\_B = B.length();

System.out.println(len\_A+len\_B);

if(A.compareTo(B)>0)

System.out.println("Yes");

else System.out.println("No");

System.out.print(((A.substring(0,1)).toUpperCase().concat(A.substring(1,len\_A)))+" "+

((B.substring(0,1)).toUpperCase().concat(B.substring(1,len\_B))));

}

}

# Java String Reverse | HackerRank Solution

Hello coders, today we are going to solve **Java String Reverse HackerRank Solution**.

## ****Problem****

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.

**Read :**[*How to reverse string in Java ?*](https://www.codingbroz.com/basic-java-program-reverse-a-string-in-java/)

#### **Constraints**

**A**will consists at most **50**lower case English letter.

##### **Sample Input**

madam

##### **Sample Output**

Yes

## ****Solution –**** ****Java String Reverse****

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String A=sc.next();

/\* Enter your code here. Print output to STDOUT. \*/

String rev="";

for(int i=A.length()-1;i>=0;i--)

{

rev = rev+A.charAt(i);

}

if(rev.equals(A))

System.out.print("Yes");

else

System.out.print("No");

}

}

# Java Anagrams | HackerRank Solution

Hello coders, today we are going to solve **Java Anagrams HackerRank Solution**.

## ****Problem****

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.

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

#### **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 <= length(a), length(b) <= 50**
* Strings **a**and **b**consists of English alphabetic characters.
* The comparison should NOT be case sensitive.

##### **Sample Input 0**

anagram

margana

##### **Sample Output 0**

Anagrams

##### **Explanation** **0**

|  |  |  |
| --- | --- | --- |
| **Character** | **Frequency:** anagram | **Frequency:** margana |
| A or a | 3 | 3 |
| G or g | 1 | 1 |
| N or n | 1 | 1 |
| M or m | 1 | 1 |
| R or r | 1 | 1 |

The two strings contain all the same letters in the same frequencies, so we print “Anagrams”.

##### **Sample Input 1**

anagramm

marganaa

##### **Sample Output 1**

Not Anagrams

##### **Explanation** **1**

|  |  |  |
| --- | --- | --- |
| **Character** | **Frequency:** anagramm | **Frequency:** marganaa |
| A or a | 3 | 4 |
| G or g | 1 | 1 |
| N or n | 1 | 1 |
| M or m | 2 | 1 |
| R or r | 1 | 1 |

The two strings don’t contain the same number of a’s and m’s, so we print “Not Anagrams”.

##### **Sample Input 2**

hello

hello

##### **Sample Output 2**

Anagrams

##### **Explanation 2**

|  |  |  |
| --- | --- | --- |
| **Character** | **Frequency:** hello | **Frequency:** hello |
| E or e | 1 | 1 |
| H or h | 1 | 1 |
| L or l | 2 | 2 |
| O or o | 1 | 1 |

The two strings contain all the same letters in the same frequencies, so we print “Anagrams”.

## ****Solution –**** ****Java Anagrams****

import java.util.Scanner;

public class Solution {

static boolean isAnagram(String a, String b) {

// Complete the function

String s1 = a;

String s2 = b;

s1=s1.toLowerCase();

s2=s2.toLowerCase();

if(s1.length()==s2.length())

{

int[] arr = new int[256];

int[] brr = new int[256];

for (int i = 0; i < s1.length(); i++) {

arr[(int) s1.charAt(i)] += 1;

brr[(int) s2.charAt(i)] += 1;

}

for (int i = 0; i < 256; i++) {

if (arr[i] != brr[i])

return false;

}

return true;

}

else

{

return false;

}

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String a = scan.next();

String b = scan.next();

scan.close();

boolean ret = isAnagram(a, b);

System.out.println( (ret) ? "Anagrams" : "Not Anagrams" );

}

}

String[] array = {"c", "a", "d", "z"};

// Sort the array

Arrays.sort(array);

// Print the sorted array

System.out.println(Arrays.toString(array));

[a, c, d, z]

import java.util.Arrays;

import java.util.Collections;

public class Main {

public static void main(String[] args) {

String[] array = {"c", "a", "d", "z"};

// Sort the array in reverse order

Arrays.sort(array, Collections.reverseOrder());

// Print the sorted array

System.out.println(Arrays.toString(array));

}

}

[z, d, c, a]

# Java String Tokens | HackerRank Solution

Hello coders, today we are going to solve **Java String Tokens**[**HackerRank**](https://www.codingbroz.com/java-string-tokens-solution/)**Solution**.

## ****Problem****

Given a string, **n**, matching the regular expression [A-Z, a-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.

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

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

## ****Input Format****

A single string, **s.**

#### **Constraints**

* **1 <= length of s <= 4.105**
* **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

##### **Explanation**

We consider a token to be a contiguous segment of alphabetic characters. There are a total of **10**such tokens in string **s**, and each token is printed in the same order in which it appears in string **s**.

## ****Solution –**** ****Java String Tokens****

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

String s = scan.nextLine();

scan.close();

// Write your code here.

s = s.trim();

if (s.length() == 0) {

System.out.println(0);

} else {

String[] strings = s.split("['!?,.\_@ ]+");

System.out.println(strings.length);

for (String str : strings)

System.out.println(str);

}

}

}

# Pattern Syntax Checker | HackerRank Solution

Hello coders, today we are going to solve **Pattern Syntax Checker**[**HackerRank**](https://www.codingbroz.com/pattern-syntax-checker-solution/)**Solution** **in Java**.

## ****Problem****

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.

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

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](https://www.codingbroz.com/pattern-syntax-checker-solution/) 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

## ****Solution –**** ****Pattern Syntax Checker**** ****in Java****

import java.util.Scanner;

import java.util.regex.\*;

public class Solution

{

public static void main(String[] args){

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

while(testCases>0){

String pattern = in.nextLine();

try {

Pattern.compile(pattern);

System.out.println("Valid");

} catch(PatternSyntaxException e) {

System.out.println("Invalid");

}

testCases--;

}

}

}

# Java Regex 2 – Duplicate Words | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-regex-2-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Regex 2 – Duplicate Words**[**HackerRank**](https://www.codingbroz.com/java-regex-2-solution/)**Solution**.

## ****Problem****

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](https://www.codingbroz.com/java-regex-2-solution/) so it will turn I love Love to To tO code into I love to code?

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

To solve this challenge, complete the following three lines:

1. Write a RegEx that will match any repeated word.
2. Complete the second [compile](https://www.codingbroz.com/java-regex-2-solution/) 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.

#### **Constraint**

* Each sentence consists of at most **104**English alphabetic letters and whitespaces.
* **1 <= n <= 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.

#### **Sample Input**

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

#### **Sample Output**

Goodbye bye world

Sam went to his business

Reya is the best player in eye game

in inthe

Hello Ab

#### **Explanation**

1. We remove the second occurrence of bye and the second and third occurrences of world from Goodbye bye bye world world world to get Goodbye bye world.
2. We remove the second occurrence of went and the second and third occurrences of to from Sam went went to to to his business to get Sam went to his business.
3. We remove the second occurrence of is, the second occurrence of the, and the second occurrence of eye from Reya is is the the best player in eye eye game to get Reya is the best player in eye game.
4. The sentence in inthe has no repeated words, so we do not modify it.
5. We remove the second occurrence of ab from Hello hello Ab aB to get Hello Ab. It’s important to note that our matching is case-insensitive, and we specifically retained the first occurrence of the matched word in our final string.

## ****Solution –**** ****Java Regex 2 – Duplicate Words****

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class DuplicateWords {

public static void main(String[] args) {

String regex = "\\b(\\w+)(?:\\W+\\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));

}

// Prints the modified sentence.

System.out.println(input);

}

in.close();

}

}

# Valid Username Regular Expression | HackerRank Solution

Hello coders, today we are going to solve **Valid Username Regular Expression HackerRank Solution** **in Java**.

## ****Problem****

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:

Enchanted by the Beautiful City near Cambodia Border - Nếm TV

* The username consists of **8**to **10**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 <= n <= 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.

#### **Sample Input**

8

Julia

Samantha

Samantha\_21

1Samantha

Samantha?10\_2A

JuliaZ007

Julia@007

\_Julia007

#### **Sample Input**

Invalid

Valid

Valid

Invalid

Invalid

Valid

Invalid

Invalid

## ****Solution –**** ****Valid Username Regular Expression**** ****in Java****

import java.util.Scanner;

class UsernameValidator {

/\*

\* Write regular expression here.

\*/

public static final String regularExpression = "^[A-Za-z]\\w{7,29}$";

}

public class Solution {

private static final Scanner scan = new Scanner(System.in);

public static void main(String[] args) {

int n = Integer.parseInt(scan.nextLine());

while (n-- != 0) {

String userName = scan.nextLine();

if (userName.matches(UsernameValidator.regularExpression)) {

System.out.println("Valid");

} else {

System.out.println("Invalid");

}

}

}

}

# Tag Content Extractor | HackerRank Solution

Hello coders, today we are going to solve **Tag Content Extractor HackerRank Solution** **in Java**.

## ****Problem****

In a tag-based language like XML or HTML, contents are enclosed between a start tag and an end tag like contents. 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 <= N <=100**
* Each line contains a maximum of **104**printable characters.
* The total number of characters in all test cases will not exceed **106**.

## ****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>

#### **Sample Output**

Nayeem loves counseling

Sanjay has no watch

So wait for a while

None

Imtiaz has a secret crush

## ****Solution –**** ****Tag Content Extractor**** ****in Java****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution{

public static void main(String[] args){

Pattern pattern = Pattern.compile("<([^>]+)>([^<]+)</\\1>");

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

while(testCases>0){

String line = in.nextLine();

Matcher m = pattern.matcher(line);

int matches = 0;

while(m.find()) {

matches++;

System.out.println(m.group(2));

}

if(matches == 0) {

System.out.println("None");

}

testCases--;

}

}

}

# Java BigDecimal | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-bigdecimal-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java BigDecimal**[**HackerRank**](https://www.codingbroz.com/java-bigdecimal-solution/)**Solution**.

## ****Problem****

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≡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 **si**.

#### **Constraints**

* **1 ≤ n ≤** **200**
* Each **si** has the 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

#### **Sample Input**

90

56.6

50

02.34

0.12

.12

0

000.000

-100

## ****Solution –**** ****Java BigDecimal****

import java.math.BigDecimal;

import java.util.\*;

class Solution{

public static void main(String []args){

//Input

Scanner sc= new Scanner(System.in);

int n=sc.nextInt();

String []s=new String[n+2];

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

s[i]=sc.next();

}

sc.close();

//Write your code here

for(int i=0;i<n;i++)

{

//inserting string values to bigdecimal

BigDecimal First=new BigDecimal(s[i]);

int index=i;

for(int j=i+1;j<n;j++)

{

//second BigDecimal to compare the first Bigdecimal

BigDecimal Second=new BigDecimal(s[j]);

//comparing if First element is greater that second element

//if the First element is greater than Second element than compareTo() returns 1

if(Second.compareTo(First)==1){

First=Second;

index=j;

}

}

//temporary variable to store s[i] value

String temp=s[i];

s[i]=s[index];

s[index]=temp;

}

//Output

for(int i=0;i<n;i++)

{

System.out.println(s[i]);

}

}

}

# Java Primality Test | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-primality-test-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Primality Test** [**HackerRank**](https://www.codingbroz.com/java-primality-test-solution/)**Solution** **in Java**.

## ****Problem****

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

## ****Solution –**** ****Java Primality Test****

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();

BigInteger bi = new BigInteger(n);

System.out.println(bi.isProbablePrime(10) ? "prime" : "not prime");

bufferedReader.close();

}

}

# Java 1D Array | HackerRank Solution

An array is a simple data structure used to store a collection of data in a contiguous block of memory. Each element in the collection is accessed using an index, and the elements are easy to find because they’re stored sequentially in memory.

Because the collection of elements in an array is stored as a big block of data, we typically use arrays when we know exactly how many pieces of data we’re going to have. For example, you might use an array to store a list of student ID numbers, or the names of state capitals. To create an array of integers named myArray that can hold four integer values, you would write the following code:

**int**[] myArray = new **int**[4];

This sets aside a block of memory that’s capable of storing 4 integers. Each integer storage cell is assigned a unique index ranging from 0 to one less than the size of the array, and each cell initially contains a 0. In the case of myArray, we can store integers at indices 0, 1, 2, and 3. Let’s say we wanted the last cell to store the number 12; to do this, we write:

myArray[3] = 12;

Similarly, we can print the contents of the last cell with the following code:

System.out.println(myArray[3]);

The code above prints the value stored at index 3 of myArray, which is 12 (the value we previously stored there). It’s important to note that while Java initializes each cell of an array of integers with a 0, not all languages do this.

## ****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 a0,a1,a2,………,an-1 from stdin and saves each integer ai 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 a0, the second value must be stored in a1, 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 ai.

## ****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

**Sample Output**

10

20

30

40

50

**Explanation**

When we save each integer to its corresponding index in a, we get a=[10,20,30,40,50]. The locked code prints each array element on a new line from left to right.

## ****Solution – Java 1D Array Hacker Rank Solution****

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

int[] a = new int[n];

for(int i=0;i<n;i++)

{

a[i]=scan.nextInt();

}

scan.close();

// Prints each sequential element in array a

for (int i = 0; i < a.length; i++) {

System.out.println(a[i]);

}

}

}

# Java 2D Array | HackerRank Solution

Hello coders, today we are going to solve **Java 2D Array**[**HackerRank**](https://www.codingbroz.com/java-2darray-solution/)**Solution**.

## ****Problem****

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.

#### **Sample Input**

1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 2 4 4 0

0 0 0 2 0 0

0 0 1 2 4 0

#### **Sample Input**

19

#### **Explanation**

The hourglass which has the largest sum is:

2 4 4

2

1 2 4

## ****Solution –**** ****Java 2D Array****

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 {

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

int[][] arr = new int[6][6];

for (int i = 0; i < 6; i++) {

String[] arrRowItems = scanner.nextLine().split(" ");

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int j = 0; j < 6; j++) {

int arrItem = Integer.parseInt(arrRowItems[j]);

arr[i][j] = arrItem;

}

}

int csum =Integer.MIN\_VALUE;

for(int i=0;i<6;i++)

{

for(int j=0;j<6;j++)

{

if(i>1 && j>1)

{

int sum=arr[i][j]+arr[i-1][j-1]+arr[i-2][j]+arr[i-2][j-1]

+arr[i-2][j-2]+arr[i][j-1]+arr[i][j-2];

//int sum=arr[i][j]+arr[i][j-1]+arr[i][j-2]+

// arr[i-2][j]+arr[i-2][j-1]+arr[i-2][j-2]+

// arr[i-1][j-1];

if(sum>csum)

{

csum=sum;

}

}

}

}

System.out.print(csum);

scanner.close();

}

}

# Java Subarray | HackerRank Solution

Hello coders, today we are going to solve **Java Subarray**[**HackerRank**](https://www.codingbroz.com/java-subarray-solution/)**Solution**.

## ****Problem****

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 = [a0,a*1***, **. . . , an-1]**.  
The second line contains **n**space-separated integers describing each respective element, **a*i***, in array **A**.

#### **Constraints**

* **1 ≤ n ≤** **100**
* **-104≤** **a*i*** **≤ 104**

## ****Output Format****

Print the number of subarrays of **A**having negative sums.

#### **Sample Input**

5

1 -2 4 -5 1

#### **Sample Output**

9

#### **Explanation**

There are nine negative subarrays of **A = [ 1, -2, 4, -5, 1]:**

1. [1:1] ⇒ -2
2. [3:3] ⇒ -5
3. [0:1] ⇒ 1 + -2 = -1
4. [2:3] ⇒ 4 + -5 = -1
5. [3:4] ⇒ -5 + 1 = -4
6. [1:3] ⇒ -2 + 4 + -5 = -3
7. [0:3] ⇒ 1 + -2 + 4 + -5 = -2
8. [1:4] ⇒ -2 + 4 + -5 + 1 = -2
9. [0:4] ⇒ 1 + -2 + 4 + -5 + 1 = -1

Thus, we print **9**on a new line.

## ****Solution –**** ****Java Subarray****

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

int n,sum=0,c=0;

Scanner sc = new Scanner(System.in);

n = sc.nextInt();

int[] a = new int[n];

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

a[i] = sc.nextInt();

}

for(int i=0;i<n;i++)

{

for(int j=i;j<n;j++)

{

sum = 0;

for(int k=i;k<=j;k++)

{

sum+=a[k];

}

if(sum<0)c++;

}

}

System.out.print(c);

}

}

# Java ArrayList | HackerRank Solution

Hello coders, today we are going to solve **Java ArrayList**[**HackerRank**](https://www.codingbroz.com/java-arraylist-solution/)**Solution**.

## ****Problem****

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 **yth**position of **xth** 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 **105**.

## ****Output Format****

In each line, output the number located in **yth**position of **xth** line. If there is no such position, just print “ERROR!”

#### **Sample Input**

5

5 41 77 74 22 44

1 12

4 37 34 36 52

0

3 20 22 33

5

1 3

3 4

3 1

4 3

5 5

#### **Sample Input**

74

52

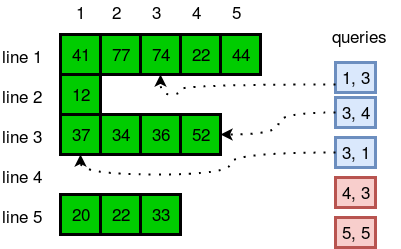
37

ERROR!

ERROR!

#### **Explanation**

The diagram below explains the queries:



## ****Solution –**** ****Java ArrayList****

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

String s = sc.nextLine();

ArrayList<ArrayList<String>> a = new ArrayList<ArrayList<String>>(n);

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

s = sc.nextLine();

a.add(i, new ArrayList<String>(Arrays.asList(s.split("\\s"))));

}

int m = sc.nextInt();

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

int x = sc.nextInt();

int y = sc.nextInt();

if (x <= a.size() && y < a.get(x-1).size() && y >= 0) {

System.out.println(a.get(x-1).get(y));

} else {

System.out.println("ERROR!");

}

}

}

}

import java.util.\*;

import java.util.ArrayList;

public class MyArrayListDemo {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        sc.nextLine(); // Consume the newline character after the integer input

        // Initialize the outer ArrayList

        List<List<String>> a = new ArrayList();

        for (int i = 0; i < n; i++) {

            String s = sc.nextLine();

            // Split the input line into words and add them as a new ArrayList

            a.add(new ArrayList<String>(Arrays.asList(s.split("\\s+"))));

        }

        System.out.println(a);

        System.out.println(a.get(0).get(3));

        System.out.println(a.get(3));

        int m = sc.nextInt();

        for (int i = 0; i < m; i++) {

            int r = sc.nextInt();

            int c = sc.nextInt();

            if (r <= a.size() && c < a.get(r-1).size() && c >= 0) {

                System.out.println(a.get(r-1).get(c));

            } else {

                System.out.println("ERROR!");

            }

        }

    }

}

# Java 1D Array (Part 2) | HackerRank Solution

Hello coders, today we are going to solve **Java 1D Array (Part 2) HackerRank Solution**.

## ****Problem****

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 <= i < n**), you can perform one of the following moves:

Traveling across Quang Binh to stunning cinematic locations

* 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 >= 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 . 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 **game0 , game1, game2, ….. , gamen-1**.

#### **Constraints**

* **1 <= q <= 5000**
* **2 <= n <= 100**
* **0 <= leap <= 100**
* It is guaranteed that the value of ***game[0]*** is always**0**.

#### **Sample Input**

STDIN Function

----- --------

4 q = 4 (number of queries)

5 3 game[] size n = 5, leap = 3 (first query)

0 0 0 0 0 game = [0, 0, 0, 0, 0]

6 5 game[] size n = 6, leap = 5 (second query)

0 0 0 1 1 1 . . .

6 3

0 0 1 1 1 0

3 1

0 1 0

#### **Sample Output**

YES

YES

NO

NO

#### **Explanation**

We perform the following **q = 4**queries:

1. For **game = [0, 0, 0, 0, 0]** and **leap = 3**, we can walk and/or jump to the end of the array because every cell contains a **0**. Because we can win, we return true.
2. For **game = [0, 0, 0, 1, 1, 1]** and **leap = 5**, we can walk to the index **1**and then jump **i + leap = 1 + 5 = 6**units to the end of the array. Because we can win, we return true.
3. For **game = [0, 0, 1, 1, 1, 0]** and **leap = 3**, there is no way for us to get past the three consecutive ones. Because we cannot win, we return false.
4. For **game = [0, 1, 0]** and **leap = 1**, there is no way for us to get past the one at index **1**. Because we cannot win, we return false.

## ****Solution –**** ****Java 1D Array (Part 2)****

import java.util.\*;

public class Solution {

public static boolean find\_path(int leap, int[] game, int x) {

if (x < 0) {

return false;

}

if (x > game.length - 1) {

return true;

}

if (game[x] != 0) {

return false;

}

game[x] = 5;

if (find\_path(leap, game, x + 1)) {

return true;

}

if (find\_path(leap, game, x + leap)) {

return true;

}

if (find\_path(leap, game, x - 1)) {

return true;

}

game[x] = 0;

return false;

}

public static boolean canWin(int leap, int[] game) {

return find\_path(leap, game, 0);

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int q = scan.nextInt();

while (q-- > 0) {

int n = scan.nextInt();

int leap = scan.nextInt();

int[] game = new int[n];

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

game[i] = scan.nextInt();

}

System.out.println( (canWin(leap, game)) ? "YES" : "NO" );

}

scan.close();

}

}

# Java List | HackerRank Solution

Hello coders, today we are going to solve **Java List HackerRank Solution**.

## ****Problem****

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

The Interesting Things About Pha Din Pass, Dien Bien That Just A Few People Know

Insert **y**at index **x :**

Insert

x y

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 **2B**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 <= N <= 4000**
* **1 <= Q <= 4000**
* Each element in is a 32-bit integer.

## ****Output Format****

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

#### **Sample Input**

5

12 0 1 78 12

2

Insert

5 23

Delete

0

#### **Sample Output**

0 1 78 12 23

#### **Explanation**

**L = [12, 0, 1, 78, 12]**

**Q0** : **Insert 23 at index 5.**

**L0** = **[12, 0, 1, 78, 12, 23]**

**Q1** : **Delete the element at index 0.**

**L1** = **[0, 1, 78, 12, 23]**

Having performed all **Q**queries, we print **L1** as a single line of space-separated integers.

## ****Solution –**** ****Java List****

import java.util.Scanner;

import java.util.LinkedList;

public class Solution {

public static void main(String[] args) {

/\* Create and fill Linked List of Integers \*/

Scanner scan = new Scanner(System.in);

int N = scan.nextInt();

LinkedList<Integer> list = new LinkedList<>();

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

int value = scan.nextInt();

list.add(value);

}

/\* Perfrom queries on Linked List \*/

int Q = scan.nextInt();

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

String action = scan.next();

if (action.equals("Insert")) {

int index = scan.nextInt();

int value = scan.nextInt();

list.add(index, value);

} else { // "Delete"

int index = scan.nextInt();

list.remove(index);

}

}

scan.close();

/\* Print our updated Linked List \*/

for (Integer num : list) {

System.out.print(num + " ");

}

}

}

# Java Map | HackerRank Solution

Hello coders, today we are going to solve **Java Map**[**HackerRank**](https://www.codingbroz.com/java-map-solution/)**Solution**.

## ****Problem****

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.

The Interesting Things About Pha Din Pass, Dien Bien That Just A Few People Know

## ****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 <= n <= 1000**

**1 <= Query <= 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

#### **Sample Output**

uncle sam=99912222

Not found

harry=12299933

## ****Solution –**** ****Java Map****

Hashmap<String,Integer> Map = new HashMap<>();

Map.put(name,phone)

Map.get(name)

Map.containsKey(s)

Map.remove(s)

Map.size()

Map.clear()

Map.keySet()

Map.values()

//Complete this code or write your own from scratch

import java.util.\*;

import java.io.\*;

class Solution {

public static void main(String [] args) throws Exception {

/\* Read input and save as entries in a HashMap \*/

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int n = Integer.valueOf(br.readLine());

HashMap<String, Integer> map = new HashMap<>();

while (n-- > 0) {

String name = br.readLine();

int phone = Integer.valueOf(br.readLine());

map.put(name, phone);

}

/\* Read each query and check if its in our HashMap \*/

String s;

while((s = br.readLine()) != null) {

if (map.containsKey(s)) {

System.out.println(s + "=" + map.get(s));

} else {

System.out.println("Not found");

}

}

br.close();

}

}

# Java Stack | HackerRank Solution

Hello coders, today we are going to solve **Java Stack**[**HackerRank**](https://www.codingbroz.com/java-stack-solution/)**Solution**.

## ****Problem****

In computer science, a stack or LIFO (last in, first out) is an abstract data type that serves as a collection of elements, with two principal operations: push, which adds an element to the collection, and pop, which removes the last element that was added.(Wikipedia)

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.

The Interesting Things About Pha Din Pass, Dien Bien That Just A Few People Know

## ****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

## ****Solution –**** ****Java Stack****

Stack.push(c);

Stack.pop();

Stack.isEmpty();

‘ char c = s.charAt(i)

import java.util.\*;

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while(sc.hasNext()) {

String input = sc.next();

System.out.println(balanaced(input));

}

}

public static boolean balanaced(String s) {

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

for(int i = 0; i < s.length(); i++) {

char c = s.charAt(i);

if(c =='[' || c == '(' || c == '{') {

stack.push(c);

}else if(c == ']') {

if(stack.isEmpty() || stack.pop() != '[') {

return false;

}

}else if(c == ')') {

if(stack.isEmpty() || stack.pop() != '(') {

return false;

}

}else if(c == '}') {

if(stack.isEmpty() || stack.pop() != '{') {

return false;

}

}

}

return stack.isEmpty();

}

}

5

5 41 77 74 22 44

1 12

4 37 34 36 52

0

3 20 22 33

5

1 3

3 4

3 1

4 3

5 5

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

{}()

({()})

{}(

[]

# Java Hashset | HackerRank Solution

Hello coders, today we are going to solve **Java Hashset HackerRank Solution**.

## ****Problem****

In [computer](https://www.codingbroz.com/java-hashset-solution/) [science](https://www.codingbroz.com/java-hashset-solution/), a set is an abstract data type that can store certain values, without any particular order, and no repeated values(Wikipedia). **{1, 2, 3}**is an example of a set ,but **{1, 2, 2}** is not a set. Today you will learn how to use sets in java by solving this problem.

You are given pairs of strings. Two pairs ***(a, b)***and ***(c, d)*** are identical if **a = c**and **b = d**. That also implies ***(a, b)*** is not same as ***(b, a)***. After taking each pair as input, you need to print number of unique pairs you currently have.

Complete the code in the editor to solve this problem.

## ****Input Format****

In the first line, there will be an integer **T**denoting number of pairs. Each of the next **T**lines will contain two strings seperated by a single space.

#### **Constraints**

* **1 <= T <= 100000**
* Length of each string is at most **5**and will consist lower case letters only.

## ****Output Format****

Print **T**lines. In the **ith**line, print number of unique pairs you have after taking **ith** pair as input.

#### **Sample Input**

5

john tom

john mary

john tom

mary anna

mary anna

#### **Sample Output**

1

2

2

3

3

#### **Explanation**

* After taking the first input, you have only one pair: (john,tom)
* After taking the second input, you have two pairs: (john, tom) and (john, mary)
* After taking the third input, you still have two unique pairs.
* After taking the fourth input, you have three unique pairs: (john,tom), (john, mary) and (mary, anna)
* After taking the fifth input, you still have three unique pairs.

## ****Solution –**** ****Java Hashset****

**HashSet<String> hash = new HashSet<String>();**

**hash.add("Volvo");**

**hash.contains("Volvo");**

**hash.remove("Volve");**

**hash.clear();**

**hash.size();**

**for (String i : hash){**

**}**

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

int t = s.nextInt();

String [] pair\_left = new String[t];

String [] pair\_right = new String[t];

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

pair\_left[i] = s.next();

pair\_right[i] = s.next();

}

//Write your code here

HashSet<String> pairs = new HashSet<String>(t);

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

{

pairs.add("(" + pair\_left[i] + ", " + pair\_right[i] + ")" );

System.out.println(pairs.size());

}

}

}

5

john tom

john mary

john tom

mary anna

mary anna

# Java Generics | HackerRank Solution

Hello coders, today we are going to solve **Java Generics**[**HackerRank**](https://www.codingbroz.com/java-generics-solution/)**Solution**.

## ****Problem****

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.

## ****Solution –**** ****Java Generics****

import java.io.IOException;

import java.lang.reflect.Method;

class Printer

{

//Write your code here

public <T> void printArray(T[] elements){

for (T element : elements){

System.out.println(element);

}

}

}

public class Solution {

public static void main( String args[] ) {

Printer myPrinter = new Printer();

Integer[] intArray = { 1, 2, 3 };

String[] stringArray = {"Hello", "World"};

myPrinter.printArray(intArray);

myPrinter.printArray(stringArray);

int count = 0;

for (Method method : Printer.class.getDeclaredMethods()) {

String name = method.getName();

if(name.equals("printArray"))

count++;

}

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

}

}

# Java Comparator | HackerRank Solution

Hello coders, today we are going to solve **Java Comparator**[**HackerRank**](https://www.codingbroz.com/java-comparator-solution/)**Solution**.

## ****Problem****

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.

#### **Constraint**

* **0 <= score <= 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.

#### **Sample Input**

5

amy 100

david 100

heraldo 50

aakansha 75

aleksa 150

#### **Sample Output**

aleksa 150

amy 100

david 100

aakansha 75

heraldo 50

## ****Solution –**** ****Java Comparator****

import java.util.\*;

// Write your Checker class here

class Checker implements Comparator {

public int compare(Object o1, Object o2) {

Player p1 = (Player) o1;

Player p2 = (Player) o2;

if (p2.score - p1.score == 0) return p1.name.compareTo(p2.name);

return p2.score - p1.score;

}

}

class Player{

String name;

int score;

Player(String name, int score){

this.name = name;

this.score = score;

}

}

class Solution {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

Player[] player = new Player[n];

Checker checker = new Checker();

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

player[i] = new Player(scan.next(), scan.nextInt());

}

scan.close();

Arrays.sort(player, checker);

for(int i = 0; i < player.length; i++){

System.out.printf("%s %s\n", player[i].name, player[i].score);

}

}

}

5

amy 100

david 100

heraldo 50

aakansha 75

aleksa 150

# Java Sort | HackerRank Solution

Hello coders, today we are going to solve **Java Sort**[**HackerRank**](https://www.codingbroz.com/java-sort-solution/)**Solution**.

## ****Problem****

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.

The Interesting Things About Pha Din Pass, Dien Bien That Just A Few People Know

**Hint :**You can use comparators to sort a list of objects.

## ****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 <= N <= 1000**
* **0 <= ID <= 100000**
* **5 <= |Name| <= 30**
* **0 <= |CGPA| <= 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.

#### **Sample Input**

5

33 Rumpa 3.68

85 Ashis 3.85

56 Samiha 3.75

19 Samara 3.75

22 Fahim 3.76

#### **Sample Output**

Ashis

Fahim

Samara

Samiha

Rumpa

## ****Solution –**** ****Java Sort****

import java.util.\*;

class Student{

private int id;

private String fname;

private double cgpa;

public Student(int id, String fname, double cgpa) {

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;

}

}

//Complete the code

public class Solution

{

public static void main(String[] args){

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

List<Student> studentList = new ArrayList<Student>();

while(testCases>0){

int id = in.nextInt();

String fname = in.next();

double cgpa = in.nextDouble();

Student st = new Student(id, fname, cgpa);

studentList.add(st);

testCases--;

}

Collections.sort(studentList, Comparator.comparing(Student :: getCgpa).reversed().thenComparing(Student :: getFname).thenComparing(Student :: getId));

for(Student st: studentList){

System.out.println(st.getFname());

}

}

}

# Java Dequeue | HackerRank Solution

Hello coders, today we are going to solve **Java Dequeue**[**HackerRank**](https://www.codingbroz.com/java-dequeue-solution/)**Solution**.

## ****Problem****

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<>();

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 <= N <= *100000***
* **1 <= M <= *100000***
* **M <= N**
* The numbers in the array will range between **[10000000]**.

## ****Output Format****

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

#### **Sample Input**

6 3

5 3 5 2 3 2

#### **Sample Output**

3

#### **Explanation**

In the sample testcase, there are 4 subarrays of contiguous numbers.

* **s1 = <5,3,5> –**Has **2**unique numbers.
* **s2 = <3,5,2> –**Has **3**unique numbers.
* **s3 = <5,2,3> –**Has **3**unique numbers.
* **s4 = <2,3,2> –**Has **2**unique numbers.

In these subarrays, there are **2,3,3,2**unique numbers, respectively. The maximum amount of unique numbers among all possible contiguous subarrays is **3**.

## ****Solution –**** ****Java Dequeue****

6 3

5 3 5 2 3 2

import java.util.\*;

public class test {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

Deque<Integer> deque = new ArrayDeque<>();

HashSet<Integer> set = new HashSet<>();

int n = in.nextInt();

int m = in.nextInt();

int max = Integer.MIN\_VALUE;

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

int input = in.nextInt();

deque.add(input);

set.add(input);

if (deque.size() == m) {

if (set.size() > max) max = set.size();

int first = deque.remove();

if (!deque.contains(first)) set.remove(first);

}

}

System.out.println(max);

}

}

# Java Priority Queue | HackerRank Solution

Hello coders, today we are going to solve **Java Priority Queue**[**HackerRank**](https://www.codingbroz.com/java-priority-queue-solution/)**Solution**.

In computer science, a priority queue is an abstract data type which is like a regular queue, but where additionally each element has a “priority” associated with it. In a priority queue, an element with high priority is served before an element with low priority. – Wikipedia

## ****Problem****

There are a number of students in a school who wait to be served. Two types of events, ENTER and SERVED, can take place which are described below.

* ENTER: A student with some priority enters the queue to be served.
* SERVED: The student with the highest priority is served (removed) from the queue.

A unique id is assigned to each student entering the queue. The queue serves the students based on the following criteria (priority criteria):

1. The student having the highest Cumulative Grade Point Average (CGPA) is served first.
2. Any students having the same CGPA will be served by name in ascending case-sensitive alphabetical order.
3. Any students having the same CGPA and name will be served in ascending order of the id.

Create the following two classes:

* The Student class should implement:
  + The constructor Student(int id, String name, double cgpa).
  + The method int getID() to return the id of the student.
  + The method String getName() to return the name of the student.
  + The method double getCGPA() to return the CGPA of the student.
* The Priorities class should implement the method List<Student> getStudents(List events) to process all the given events and return all the students yet to be served in the priority order.

## ****Input Format****

The first line contains an integer, **n**, describing the total number of events. Each of the **n**subsequent lines will be of the following two forms:

* ENTER name CGPA id: The student to be inserted into the priority queue.
* SERVED: The highest priority student in the queue was served.

The locked stub code in the editor reads the input and tests the correctness of the Student and Priorities classes implementation.

#### **Constraints**

* **2 <= n <= 1000**
* **0 <= CGPA <= 4.00**
* **1 <= id <= 105**
* **2 <= |name| <= 30**

## ****Output****

The locked stub code prints the names of the students yet to be served in the priority order. If there are no such student, then the code prints EMPTY.

#### **Sample Input**

12

ENTER John 3.75 50

ENTER Mark 3.8 24

ENTER Shafaet 3.7 35

SERVED

SERVED

ENTER Samiha 3.85 36

SERVED

ENTER Ashley 3.9 42

ENTER Maria 3.6 46

ENTER Anik 3.95 49

ENTER Dan 3.95 50

SERVED

#### **Sample Output**

Dan

Ashley

Shafaet

Mariat

#### **Explanation**

In this case, the number of events is 12. Let the name of the queue be Q.

* John is added to Q. So, it contains (John, 3.75, 50).
* Mark is added to Q. So, it contains (John, 3.75, 50) and (Mark, 3.8, 24).
* Shafaet is added to Q. So, it contains (John, 3.75, 50), (Mark, 3.8, 24), and (Shafaet, 3.7, 35).
* Mark is served as he has the highest CGPA. So, Q contains (John, 3.75, 50) and (Shafaet, 3.7, 35).
* John is served next as he has the highest CGPA. So, Q contains (Shafaet, 3.7, 35).
* Samiha is added to Q. So, it contains (Shafaet, 3.7, 35) and (Samiha, 3.85, 36).
* Samiha is served as she has the highest CGPA. So, Q contains (Shafaet, 3.7, 35).
* Now, four more students are added to Q. So, it contains (Shafaet, 3.7, 35), (Ashley, 3.9, 42), (Maria, 3.6, 46), (Anik, 3.95, 49), and (Dan, 3.95, 50).
* Anik is served because though both Anil and Dan have the highest CGPA but Anik comes first when sorted in alphabetic order. So, Q contains (Dan, 3.95, 50), (Ashley, 3.9, 42), (Shafaet, 3.7, 35), and (Maria, 3.6, 46).

As all events are completed, the name of each of the remaining students is printed on a new line.

## ****Solution –**** ****Java Priority Queue****

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

// import Priority Queue Collection Framework

import java.util.\*;

/\*

\* Create the Student and Priorities classes here.

\*/

class Student implements Comparable<Student>{

String name = new String();

double cgpa;

int id;

public Student(String name,double cgpa,int id)

{

this.name = name;

this.cgpa = cgpa;

this.id = id;

}

public String getName(){

return this.name;

}

public int compareTo(Student s)

{

if(cgpa == s.cgpa)

{

if(name.compareTo(s.name) == 0)

{

if(id == s.id)

return 0;

else if (id > s.id)

return 1;

else

return -1;

}

else

return name.compareTo(s.name);

}

else if(cgpa > s.cgpa)

return -1;

else

return 1;

}

}

class Priorities{

public ArrayList<Student> getStudents(List<String> events)

{

int n = events.size();

PriorityQueue<Student> pq = new PriorityQueue<Student>();

for(String i:events)

{

String[] s = new String[4];

s = i.split("\\s");

if(s.length>1)

{

pq.add(new Student(s[1],Double.valueOf(s[2]),Integer.valueOf(s[3])));

}

else

{

pq.poll();

}

}

while(pq.size()>1)

{

System.out.println(pq.poll().name);

}

return new ArrayList<Student>(pq);

}

}

public class Solution {

private final static Scanner scan = new Scanner(System.in);

private final static Priorities priorities = new Priorities();

public static void main(String[] args) {

int totalEvents = Integer.parseInt(scan.nextLine());

List<String> events = new ArrayList<>();

while (totalEvents-- != 0) {

String event = scan.nextLine();

events.add(event);

}

List<Student> students = priorities.getStudents(events);

if (students.isEmpty()) {

System.out.println("EMPTY");

} else {

for (Student st: students) {

System.out.println(st.getName());

}

}

}

}

# Java Inheritance I | HackerRank Solution

Hello coders, today we are going to solve **Java Inheritance 1 HackerRank Solution**.

## ****Problem****

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:

I am walking

I am flying

I am singing

## ****Solution –**** ****Java Inheritance 1****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Animal{

void walk()

{

System.out.println("I am walking");

}

}

class Bird extends Animal

{

void fly()

{

System.out.println("I am flying");

}

void sing()

{

System.out.println("I am singing");

}

}

public class Solution{

public static void main(String args[]){

Bird bird = new Bird();

bird.walk();

bird.fly();

bird.sing();

}

}

# Java Inheritance II | HackerRank Solution

Hello coders, today we are going to solve **Java Inheritance 2**[**HackerRank**](https://www.codingbroz.com/java-inheritance-2-solution/)**Solution**.

## ****Problem****

Write the following code in your editor below:

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

Your classes should not be be.

## ****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

## ****Solution – Java Inheritance 2****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

//Write your code here

class Arithmetic

{

int add(int a,int b)

{

return a+b;

}

}

class Adder extends Arithmetic {}

public class Solution{

public static void main(String []args){

// Create a new Adder object

Adder a = new Adder();

// Print the name of the superclass on a new line

System.out.println("My superclass is: " + a.getClass().getSuperclass().getName());

// Print the result of 3 calls to Adder's `add(int,int)` method as 3 space-separated integers:

System.out.print(a.add(10,32) + " " + a.add(10,3) + " " + a.add(10,10) + "\n");

}

}

# Java Abstract Class | HackerRank Solution

Hello coders, today we are going to solve **Java Abstract Class**[**HackerRank**](https://www.codingbroz.com/java-abstract-class-solution/)**Solution**.

## ****Problem****

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

## ****Solution – Java Abstract Class****

import java.util.\*;

abstract class Book{

String title;

abstract void setTitle(String s);

String getTitle(){

return title;

}

}

//Write MyBook class here

class MyBook extends Book {

@Override

void setTitle(String s) {

title = s;

}

}

public class Main{

public static void main(String []args){

//Book new\_novel=new Book(); This line prHMain.java:25: error: Book is abstract; cannot be instantiated

Scanner sc=new Scanner(System.in);

String title=sc.nextLine();

MyBook new\_novel=new MyBook();

new\_novel.setTitle(title);

System.out.println("The title is: "+new\_novel.getTitle());

sc.close();

}

}

# Java Interface | HackerRank Solution

Hello coders, today we are going to solve **Java Interface**[**HackerRank**](https://www.codingbroz.com/java-interface-solution/)**Solution**.

## ****Problem****

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 Advanced Arithmetic 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.

## ****Solution – Java Interface****

import java.util.\*;

interface AdvancedArithmetic{

int divisor\_sum(int n);

}

//Write your code here

class MyCalculator implements AdvancedArithmetic {

public int divisor\_sum(int n) {

int sum=0;

for(int i=1;i<=n;i++) {

if(n%i==0)

sum+=i;

}

return sum;

}

}

class Solution{

public static void main(String []args){

MyCalculator my\_calculator = new MyCalculator();

System.out.print("I implemented: ");

ImplementedInterfaceNames(my\_calculator);

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

System.out.print(my\_calculator.divisor\_sum(n) + "\n");

sc.close();

}

/\*

\* ImplementedInterfaceNames method takes an object and prints the name of the interfaces it implemented

\*/

static void ImplementedInterfaceNames(Object o){

Class[] theInterfaces = o.getClass().getInterfaces();

for (int i = 0; i < theInterfaces.length; i++){

String interfaceName = theInterfaces[i].getName();

System.out.println(interfaceName);

}

}

}

# Java Method Overriding | HackerRank Solution

Hello coders, today we are going to solve **Java Method Overriding**[**HackerRank**](https://www.codingbroz.com/java-method-overriding-solution/)**Solution**.

## ****Problem****

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 with (the number of players on a Soccer team).

## ****Output Format****

When executed, your completed code should print the following:

Generic Sports

Each team has n players in Generic Sports

Soccer Class

Each team has 11 players in Soccer Class

## ****Solution – Java Method Overriding****

import java.util.\*;

class Sports{

String getName(){

return "Generic Sports";

}

void getNumberOfTeamMembers(){

System.out.println( "Each team has n players in " + getName() );

}

}

class Soccer extends Sports{

@Override

String getName(){

return "Soccer Class";

}

// Write your overridden getNumberOfTeamMembers method here

@Override

void getNumberOfTeamMembers () {

System.out.println( "Each team has 11 players in " + getName() );

}

}

public class Solution{

public static void main(String []args){

Sports c1 = new Sports();

Soccer c2 = new Soccer();

System.out.println(c1.getName());

c1.getNumberOfTeamMembers();

System.out.println(c2.getName());

c2.getNumberOfTeamMembers();

}

}

# Java Method Overriding 2 (Super Keyword) | HackerRank Solution

Hello coders, today we are going to solve **Java Method Overriding 2 (Super Keyword) HackerRank Solution**.

## ****Problem****

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.

#### **Super Keyword in java**

* The super keyword in java is a reference variable which is used to refer immediate parent class object.
* Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

#### **Usage of Java Super Keyword**

* super can be used to refer immediate parent class instance variable.
* super can be used to invoke immediate parent class method.
* super() can be used to invoke immediate parent class constructor.

## ****Solution – Java Method Overriding 2 (Super Keyword)****

import java.util.\*;

import java.io.\*;

class BiCycle{

String define\_me(){

return "a vehicle with pedals.";

}

}

class MotorCycle extends BiCycle{

String define\_me(){

return "a cycle with an engine.";

}

MotorCycle(){

System.out.println("Hello I am a motorcycle, I am "+ define\_me());

// String temp=define\_me(); //Fix this line

String temp=super.define\_me(); //Fixed

System.out.println("My ancestor is a cycle who is "+ temp );

}

}

class Solution{

public static void main(String []args){

MotorCycle M=new MotorCycle();

}

}

# Java Instanceof keyword | HackerRank Solution

Hello coders, today we are going to solve **Java Instanceof keyword**[**HackerRank**](https://www.codingbroz.com/java-instanceof-keyword-solution/)**Solution**.

## ****Problem****

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

## ****Solution – Java Instanceof keyword****

import java.util.\*;

class Student{}

class Rockstar{ }

class Hacker{}

public class InstanceOFTutorial{

static String count(ArrayList mylist){

int a = 0,b = 0,c = 0;

for(int i = 0; i < mylist.size(); i++){

Object element=mylist.get(i);

if(element instanceof Student )

a++;

if(element instanceof Rockstar)

b++;

if(element instanceof Hacker)

c++;

}

String ret = Integer.toString(a)+" "+ Integer.toString(b)+" "+ Integer.toString(c);

return ret;

}

public static void main(String []args){

ArrayList mylist = new ArrayList();

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

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

String s=sc.next();

if(s.equals("Student"))mylist.add(new Student());

if(s.equals("Rockstar"))mylist.add(new Rockstar());

if(s.equals("Hacker"))mylist.add(new Hacker());

}

System.out.println(count(mylist));

}

}

# Java Iterator | HackerRank Solution

Hello coders, today we are going to solve **Java Iterator**[**HackerRank**](https://www.codingbroz.com/java-iterator-solution/)**Solution**.

## ****Problem****

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

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.

## ****Solution – Java Iterator****

import java.util.\*;

public class Main{

static Iterator func(ArrayList mylist){

Iterator it=mylist.iterator();

while(it.hasNext()){

import java.util.\*;

public class Main{

static Iterator func(ArrayList mylist){

Iterator it=mylist.iterator();

while(it.hasNext()){

Object element = it.next();

if(element instanceof String)//Hints: use instanceof operator

break;

}

return it;

}

@SuppressWarnings({ "unchecked" })

public static void main(String []args){

ArrayList mylist = new ArrayList();

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int m = sc.nextInt();

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

mylist.add(sc.nextInt());

}

mylist.add("###");

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

mylist.add(sc.next());

}

Iterator it=func(mylist);

while(it.hasNext()){

Object element = it.next();

System.out.println((String)element);

}

}

}

break;

}

return it;

}

@SuppressWarnings({ "unchecked" })

public static void main(String []args){

ArrayList mylist = new ArrayList();

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int m = sc.nextInt();

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

mylist.add(sc.nextInt());

}

mylist.add("###");

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

mylist.add(sc.next());

}

Iterator it=func(mylist);

while(it.hasNext()){

Object element = it.next();

System.out.println((String)element);

}

}

}

/\* Java Iterator

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

import java.util.\*;

public class Example

{

public static void main(String []argh)

{

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.

\*/

import java.util.\*;

public class JavaIterator

{

static Iterator func(ArrayList mylist)

{

Iterator it=mylist.iterator();

while(it.hasNext())

{

Object element = it.next();

if(element instanceof String) //Hints: use instanceof operator

break;

}

return it;

}

public static void main(String []argh)

{

ArrayList mylist = new ArrayList();

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

for(int i=0;i<n;i++)

{

mylist.add(sc.nextInt());

}

mylist.add("###");

for(int i=0;i<m;i++)

{

mylist.add(sc.next());

}

Iterator it=func(mylist);

while(it.hasNext())

{

Object element = it.next();

System.out.println((String)element);

}

}

}

# Java Exception Handling (Try-catch) | HackerRank Solution

Hello coders, today we are going to solve **Java Exception Handling (Try-catch) HackerRank Solution**.

## ****Problem****

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

#### **Sample Input** **2**

10

0

#### **Sample Output** **2**

java.lang.ArithmeticException: / by zero

#### **Sample Input** **3**

23.323

0

#### **Sample Output** **3**

java.util.InputMismatchException

## ****Solution – Java Exception Handling (Try-catch)****

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

try{

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();

int b = sc.nextInt();

int c = a/b;

System.out.print(c);

}

catch(InputMismatchException ob){

System.out.print("java.util.InputMismatchException");

}

catch(Exception e)

{

System.out.print(e);

}

}

}

InputMismatchException

# Java Exception Handling | HackerRank Solution

Hello coders, today we are going to solve **Java Exception Handling**[**HackerRank**](https://www.codingbroz.com/java-exception-handling-solution/)**Solution**.

## ****Problem****

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 **np**. 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 or 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.

#### **Constraint**

* **-10 <= n <= 10**
* **-10 <= p <= 10**

## ****Output Format****

Each line of the output contains the result **np**, 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.

#### **Sample Input**

3 5

2 4

0 0

-1 -2

-1 3

#### **Sample Output**

243

16

java.lang.Exception: n and p should not be zero.

java.lang.Exception: n or p should not be negative.

java.lang.Exception: n or p should not be negative.

#### **Explanation**

* In the first two cases, both **n**and **p**are postive. So, the power function returns the answer correctly.
* In the third case, both **n**and **p**are zero. So, the exception, “n and p should not be zero.”, is printed.
* In the last two cases, at least one out of **n**and **p**is negative. So, the exception, “n or p should not be negative.”, is printed for these two cases.

## ****Solution – Java Exception Handling****

import java.util.Scanner;

import java.math.\*;

class MyCalculator {

/\*

\* Create the method long power(int, int) here.

\*/

public static long power(int a,int b) throws Exception{

long la = a;

long lb = b;

long c = (long)Math.pow(la,lb);

if(la==0 && lb==0) {

throw new Exception("n and p should not be zero.");

}

else if(la<0 || lb <0) {

throw new Exception("n or p should not be negative.");

}

else

return c;

}

}

public class Solution {

public static final MyCalculator my\_calculator = new MyCalculator();

public static final Scanner in = new Scanner(System.in);

public static void main(String[] args) {

while (in .hasNextInt()) {

int n = in .nextInt();

int p = in .nextInt();

try {

System.out.println(my\_calculator.power(n, p));

} catch (Exception e) {

System.out.println(e);

}

}

}

}

# Java Varargs – Simple Addition | HackerRank Solution

Hello coders, today we are going to solve **Java Varargs – Simple Addition**[**HackerRank**](https://www.codingbroz.com/java-varargs-simple-addition-solution/)**Solution**.

## ****Problem****

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

## ****Solution – Java Varargs – Simple Addition****

import java.io.\*;

import java.lang.reflect.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

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);

}

}

public class Solution {

public static void main(String[] args) {

try{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int n1=Integer.parseInt(br.readLine());

int n2=Integer.parseInt(br.readLine());

int n3=Integer.parseInt(br.readLine());

int n4=Integer.parseInt(br.readLine());

int n5=Integer.parseInt(br.readLine());

int n6=Integer.parseInt(br.readLine());

Add ob=new Add();

ob.add(n1,n2);

ob.add(n1,n2,n3);

ob.add(n1,n2,n3,n4,n5);

ob.add(n1,n2,n3,n4,n5,n6);

Method[] methods=Add.class.getDeclaredMethods();

Set<String> set=new HashSet<>();

boolean overload=false;

for(int i=0;i<methods.length;i++)

{

if(set.contains(methods[i].getName()))

{

overload=true;

break;

}

set.add(methods[i].getName());

}

if(overload)

{

throw new Exception("Overloading not allowed");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

# Java Reflection – Attributes | HackerRank Solution

Hello coders, today we are going to solve **Java Reflection – Attributes**[**HackerRank**](https://www.codingbroz.com/java-reflection-attributes-solution/)**Solution**.

## ****Problem****

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.

## ****Solution – Java Reflection – Attributes****

public class Solution {

public static void main(String[] args){

Class student = Student.class;

Method[] methods = student.getDeclaredMethods();

ArrayList<String> methodList = new ArrayList<>();

for(Method m : methods){

methodList.add(m.getName());

}

Collections.sort(methodList);

for(String name: methodList){

System.out.println(name);

}

}

}

# Can You Access? | HackerRank Solution

Hello coders, today we are going to solve **Can You Access?**[**HackerRank**](https://www.codingbroz.com/can-you-access-solution/)**Solution** **in Java**.

## ****Problem****

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 . You have to call the method powerof2 of the class Inner.Private from the main method of the class Solution.

#### **Constraints**

**1 <= num <= 230**

#### **Sample Input**

8

#### **Sample Output**

8 is power of 2

An instance of class: Solution.Inner.Private has been created

## ****Solution – Can You Access?**** ****in Java****

import java.io.\*;

import java.lang.reflect.\*;

import java.util.\*;

import java.util.regex.\*;

import java.security.\*;

public class Solution {

public static void main(String[] args) throws Exception {

DoNotTerminate.forbidExit();

try{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int num = Integer.parseInt(br.readLine().trim());

Object o;// Must be used to hold the reference of the instance of the class Solution.Inner.Private

System.out.println(num + " is " + ((Inner.Private) (o = new Inner().new Private())).powerof2(num));

System.out.println("An instance of class: " + o.getClass().getCanonicalName() + " has been created");

}//end of try

catch (DoNotTerminate.ExitTrappedException e) {

System.out.println("Unsuccessful Termination!!");

}

}//end of main

static class Inner{

private class Private{

private String powerof2(int num){

return ((num&num-1)==0)?"power of 2":"not a power of 2";

}

}

}//end of Inner

}//end of Solution

class DoNotTerminate { //This class prevents exit(0)

public static class ExitTrappedException extends SecurityException {

private static final long serialVersionUID = 1L;

}

public static void forbidExit() {

final SecurityManager securityManager = new SecurityManager() {

@Override

public void checkPermission(Permission permission) {

if (permission.getName().contains("exitVM")) {

throw new ExitTrappedException();

}

}

};

System.setSecurityManager(securityManager);

}

}

# Prime Checker | HackerRank Solution

Hello coders, today we are going to solve **Prime Checker**[**HackerRank**](https://www.codingbroz.com/prime-checker-solution/)**Solution** **in Java**.

## ****Problem****

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

## ****Solution – Prime Checker**** ****in Java****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

import java.lang.reflect.\*;

import static java.lang.System.in;

class Prime {

void checkPrime(int... numbers) {

for (int num : numbers) {

if (isPrime(num)) {

System.out.print(num + " ");

}

}

System.out.println();

}

boolean isPrime(int n) {

if (n < 2) {

return false;

} else if (n == 2) {

return true;

} else if (n % 2 == 0) {

return false;

}

int sqrt = (int) Math.sqrt(n);

for (int i = 3; i <= sqrt; i += 2) {

if (n % i == 0) {

return false;

}

}

return true;

}

}

public class Solution {

public static void main(String[] args) {

try{

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

int n1=Integer.parseInt(br.readLine());

int n2=Integer.parseInt(br.readLine());

int n3=Integer.parseInt(br.readLine());

int n4=Integer.parseInt(br.readLine());

int n5=Integer.parseInt(br.readLine());

Prime ob=new Prime();

ob.checkPrime(n1);

ob.checkPrime(n1,n2);

ob.checkPrime(n1,n2,n3);

ob.checkPrime(n1,n2,n3,n4,n5);

Method[] methods=Prime.class.getDeclaredMethods();

Set<String> set=new HashSet<>();

boolean overload=false;

for(int i=0;i<methods.length;i++)

{

if(set.contains(methods[i].getName()))

{

overload=true;

break;

}

set.add(methods[i].getName());

}

if(overload)

{

throw new Exception("Overloading not allowed");

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

//Song ans:

import static java.lang.System.in;

class Prime{

    void checkPrime(int... intArg){

        for (int i : intArg) {

            if(isPrime(i)){

                System.out.print(i + " ");

            }

        }

        System.out.println();

    }

    boolean isPrime(int num){

        if(num<2)

            return false;

        else{

            for(int i=2; i< num; i++){

                if(num%i==0)

                    return false;

            }

        }

        return true;

    }

}

# Java Factory Pattern | HackerRank Solution

Hello coders, today we are going to solve **Java Factory Pattern**[**HackerRank**](https://www.codingbroz.com/java-factory-pattern-solution/)**Solution**.

## ****Problem****

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!

## ****Solution – Java Factory Pattern****

import java.util.\*;

import java.security.\*;

interface Food {

public String getType();

}

class Pizza implements Food {

public String getType() {

return "Someone ordered a Fast Food!";

}

}

class Cake implements Food {

public String getType() {

return "Someone ordered a Dessert!";

}

}

class FoodFactory {

public Food getFood(String order) {

switch (order){

case "pizza": return new Pizza();

case "cake" : return new Cake();

default : return null;

}

}//End of getFood method

}//End of factory class

public class Solution {

public static void main(String args[]){

Do\_Not\_Terminate.forbidExit();

try{

Scanner sc=new Scanner(System.in);

//creating the factory

FoodFactory foodFactory = new FoodFactory();

//factory instantiates an object

Food food = foodFactory.getFood(sc.nextLine());

System.out.println("The factory returned "+food.getClass());

System.out.println(food.getType());

}

catch (Do\_Not\_Terminate.ExitTrappedException e) {

System.out.println("Unsuccessful Termination!!");

}

}

}

class Do\_Not\_Terminate {

public static class ExitTrappedException extends SecurityException {

private static final long serialVersionUID = 1L;

}

public static void forbidExit() {

final SecurityManager securityManager = new SecurityManager() {

@Override

public void checkPermission(Permission permission) {

if (permission.getName().contains("exitVM")) {

throw new ExitTrappedException();

}

}

};

System.setSecurityManager(securityManager);

}

}

# Java Singleton Pattern | HackerRank Solution

Hello coders, today we are going to solve **Java Singleton Pattern HackerRank Solution**.

## ****Problem****

"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:

* A private Singleton non parameterized constructor.
* A public String instance variable named **str**.
* 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

## ****Solution – Java Singleton Pattern****

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

import java.lang.reflect.\*;

class Singleton {

private volatile static Singleton instance;

public static String str;

private Singleton() {}

static Singleton getSingleInstance() {

if (instance == null) {

synchronized (Singleton.class) {

if (instance == null) {

instance = new Singleton();

}

}

}

return instance;

}

}

# Java Visitor Pattern | HackerRank Solution

Hello coders, today we are going to solve **Java Visitor Pattern**[**HackerRank**](https://www.codingbroz.com/java-visitor-pattern-solution/)**Solution**.

## ****Problem****

**Note:** In this problem you must NOT generate any output on your own. Any such solution will be considered as being against the rules and its author will be disqualified. The output of your solution must be generated by the uneditable code provided for you in the solution template.

An important concept in Object-Oriented Programming is the open/closed principle, which means writing code that is open to extension but closed to modification. In other words, new functionality should be added by writing an extension for the existing code rather than modifying it and potentially breaking other code that uses it. This challenge simulates a real-life problem where the open/closed principle can and should be applied.

A Tree class implementing a rooted tree is provided in the editor. It has the following publicly available methods:

* getValue(): Returns the value stored in the node.
* getColor(): Returns the color of the node.
* getDepth(): Returns the depth of the node. Recall that the depth of a node is the number of edges between the node and the tree’s root, so the tree’s root has depth **0**and each descendant node’s depth is equal to the depth of its parent node **+1**.

In this challenge, we treat the internal implementation of the tree as being closed to modification, so we cannot directly modify it; however, as with real-world situations, the implementation is written in such a way that it allows external classes to extend and build upon its functionality. More specifically, it allows objects of the TreeVis class (a Visitor Design Pattern) to visit the tree and traverse the tree structure via the accept method.

There are two parts to this challenge.

## ****Part I: Implement Three Different Visitors****

Each class has three methods you must write implementations for:

1. getResult(): Return an integer denoting the **result**, which is different for each class:
   * The SumInLeavesVisitor implementation must return the sum of the values in the tree’s leaves only.
   * The ProductRedNodesVisitor implementation must return the product of values stored in all red nodes, including leaves, computed modulo **109+ 7**. Note that the product of zero values is equal to **1**.
   * The FancyVisitor implementation must return the absolute difference between the sum of values stored in the tree’s non-leaf nodes at even depth and the sum of values stored in the tree’s green leaf nodes. Recall that zero is an even number.
2. visitNode(TreeNode node): Implement the logic responsible for visiting the tree’s non-leaf nodes such that the getResult method returns the correct **result**for the implementing class’ visitor.
3. visitLeaf(TreeLeaf leaf): Implement the logic responsible for visiting the tree’s leaf nodes such that the getResult method returns the correct **result** for the implementing class’ visitor.

## ****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 (**x1**, **x2**, . . . , **xn**), a list of node colors (**c1**, **c2**, . . . , **cn**), 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 **x1**, **x2**, . . . , **xn**.  
The third line contains **n** space-separated binary integers describing the respective values of **c1**, **c2**, . . . , **cn**. Each **ci** denotes the color of the **ith**node, where **0**denotes red and **1**denotes green.  
Each of the **n-1**subsequent lines contains two space-separated integers, **ui** and **vi**, describing an edge between nodes **ui** and **vi**.

#### **Constraints**

* **2 <= n <= 105**
* **1 <= xi <= 103**
* **ci** **∈ {** **0,1 }**
* **1 <= vi** , **ui** **<= 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 **109+7**.

#### **Sample Input**

5

4 7 2 5 12

0 1 0 0 1

1 2

1 3

3 4

3 5

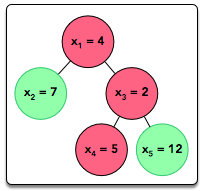
#### **Sample Output**

24

40

15

#### **Explanation**



Locked stub code in the editor tests your three class implementations as follows:

1. Creates a SumInLeavesVisitor object whose getResult method returns the sum of the leaves in the tree, which is **7 + 5 + 12 = 24**. The locked stub code prints the returned value on a new line.
2. Creates a ProductOfRedNodesVisitor object whose getResult method returns the product of the red nodes, which is **4 . 2 . 5 = 40**. The locked stub code prints the returned value on a new line.
3. Creates a FancyVisitor object whose getResult method returns the absolute difference between the sum of the values of non-leaf nodes at even depth and the sum of the values of green leaf nodes, which is | 4-(7+12) |. The locked stub code prints the returned value on a new line.

## ****Solution – Java Visitor Pattern****

import java.util.ArrayList;

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

import java.util.ArrayList;

import java.util.Scanner;

enum Color {

RED, GREEN

}

abstract class Tree {

private int value;

private Color color;

private int depth;

public Tree(int value, Color color, int depth) {

this.value = value;

this.color = color;

this.depth = depth;

}

public int getValue() {

return value;

}

public Color getColor() {

return color;

}

public int getDepth() {

return depth;

}

public abstract void accept(TreeVis visitor);

}

class TreeNode extends Tree {

private ArrayList<Tree> children = new ArrayList<>();

public TreeNode(int value, Color color, int depth) {

super(value, color, depth);

}

public void accept(TreeVis visitor) {

visitor.visitNode(this);

for (Tree child : children) {

child.accept(visitor);

}

}

public void addChild(Tree child) {

children.add(child);

}

}

class TreeLeaf extends Tree {

public TreeLeaf(int value, Color color, int depth) {

super(value, color, depth);

}

public void accept(TreeVis visitor) {

visitor.visitLeaf(this);

}

}

abstract class TreeVis

{

public abstract int getResult();

public abstract void visitNode(TreeNode node);

public abstract void visitLeaf(TreeLeaf leaf);

}

class SumInLeavesVisitor extends TreeVis {

int sumInLeaves = 0;

public int getResult() {

return sumInLeaves;

}

public void visitNode(TreeNode node) {

// empty return

}

public void visitLeaf(TreeLeaf leaf) {

sumInLeaves += leaf.getValue();

}

}

class ProductOfRedNodesVisitor extends TreeVis {

long productOfRedNodes = 1L;

public int getResult() {

return (int) (productOfRedNodes);

}

void multiply(Tree tree) {

if (tree.getColor() == Color.RED)

productOfRedNodes = (productOfRedNodes \* tree.getValue()) % (1000000007);

}

public void visitNode(TreeNode node) {

multiply(node);

}

public void visitLeaf(TreeLeaf leaf) {

multiply(leaf);

}

}

class FancyVisitor extends TreeVis {

int sumOfValuesNonLeafEvenDepth = 0;

int sumOfValuesGreenLeaf = 0;

public int getResult() {

return Math.abs(sumOfValuesGreenLeaf - sumOfValuesNonLeafEvenDepth);

}

public void visitNode(TreeNode node) {

if (node.getDepth() % 2 != 0) return;

sumOfValuesNonLeafEvenDepth += node.getValue();

}

public void visitLeaf(TreeLeaf leaf) {

if (leaf.getColor() != Color.GREEN) return;

sumOfValuesGreenLeaf += leaf.getValue();

}

}

public class Solution {

static Map<Integer, Tree> tree = new HashMap<>();

public static Tree solve() {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

Map<Integer, Object[]> nodeAtts = new HashMap<Integer, Object[]>();

for (int i = 0; i < n; i++)

nodeAtts.put(i + 1, new Object[]{sc.nextInt(), null});

for (int i = 0; i < n; i++)

nodeAtts.get(i + 1)[1] = sc.nextInt() == 0 ? Color.RED : Color.GREEN;

Map<Integer, ArrayList<Integer>> edges = new HashMap<Integer, ArrayList<Integer>>();

for (int i = 1; i <= n; i++)

edges.put(i, new ArrayList<Integer>());

for (int i = 1; i < n; i++) {

int u = sc.nextInt();

int v = sc.nextInt();

edges.get(u).add(v);

edges.get(v).add(u);

}

Tree root = new TreeNode((Integer) nodeAtts.get(1)[0], (Color) nodeAtts.get(1)[1], 0);

tree.put(1, root);

DFS(n, edges, nodeAtts);

return tree.get(1);

}

private static void DFS(int n, Map<Integer, ArrayList<Integer>> edges, Map<Integer, Object[]> nodeAtts) {

boolean[] visited = new boolean[n + 1];

TreeNode parent = (TreeNode) tree.get(1);

DFSUtil(parent, 1, visited, edges, nodeAtts);

}

private static void DFSUtil(TreeNode parent, int v, boolean[] visited, Map<Integer, ArrayList<Integer>> edges, Map<Integer, Object[]> nodeAtts) {

visited[v] = true;

if (edges.get(v).size() == 1 && v != 1) {

TreeLeaf treeLeaf = new TreeLeaf((Integer) nodeAtts.get(v)[0], (Color) nodeAtts.get(v)[1], parent.getDepth() + 1);

parent.addChild(treeLeaf);

tree.put(v, treeLeaf);

return;

}

TreeNode treeNode;

if (v != 1) {

treeNode = new TreeNode((Integer) nodeAtts.get(v)[0], (Color) nodeAtts.get(v)[1], parent.getDepth() + 1);

parent.addChild(treeNode);

tree.put(v, treeNode);

} else

treeNode = (TreeNode) tree.get(1);

Iterator<Integer> iterator = edges.get(v).iterator();

while (iterator.hasNext()) {

int n = iterator.next();

if (!visited[n]) {

DFSUtil(treeNode, n, visited, edges, nodeAtts);

}

}

}

public static void main(String[] args) {

Tree root = solve();

SumInLeavesVisitor vis1 = new SumInLeavesVisitor();

ProductOfRedNodesVisitor vis2 = new ProductOfRedNodesVisitor();

FancyVisitor vis3 = new FancyVisitor();

root.accept(vis1);

root.accept(vis2);

root.accept(vis3);

int res1 = vis1.getResult();

int res2 = vis2.getResult();

int res3 = vis3.getResult();

System.out.println(res1);

System.out.println(res2);

System.out.println(res3);

}

}

# Java Annotations | HackerRank Solution

Hello coders, today we are going to solve **Java Annotations**[**HackerRank**](https://www.codingbroz.com/java-annotations-solution/)**Solution**.

## ****Problem****

Java annotation can be used to define the metadata of a Java class or class element. We can use Java annotation at the [compile](https://www.codingbroz.com/java-annotations-solution/) time to instruct the [compiler](https://www.codingbroz.com/java-annotations-solution/) about the build process. Annotation is also used at runtime to get insight into the properties of class elements.

Java annotation can be added to an element in the following way:

@Entity

Class DemoClass{

}

We can also set a value to the annotation member. For example:

@Entity(EntityName="DemoClass")

Class DemoClass{

}

In Java, there are several built-in annotations. You can also define your own annotations in the following way:

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@interface FamilyBudget {

String userRole() default "GUEST";

}

Here, we define an annotation **FamilyBudget**, where **userRole**is the only member in that custom annotation. The **userRole** takes only **String**type values, and the default is **“GUEST”**. If we do not define the value for this annotation member, then it takes the default. By using **@Target**, we can specify where our annotation can be used. For example, the **FamilyBudget** annotation can only be used with the method in a class. **@Retention** defines whether the annotation is available at runtime. To learn more about Java annotation.

Take a look at the following code segment:

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@interface FamilyBudget {

String userRole() default "GUEST";

}

class FamilyMember {

public void seniorMember(int budget, int moneySpend) {

System.out.println("Senior Member");

System.out.println("Spend: " + moneySpend);

System.out.println("Budget Left: " + (budget - moneySpend));

}

public void juniorUser(int budget, int moneySpend) {

System.out.println("Junior Member");

System.out.println("Spend: " + moneySpend);

System.out.println("Budget Left: " + (budget - moneySpend));

}

}

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

while (testCases > 0) {

String role = in.next();

int spend = in.nextInt();

try {

Class annotatedClass = FamilyMember.class;

Method[] methods = annotatedClass.getMethods();

for (Method method : methods) {

if (method.isAnnotationPresent(FamilyBudget.class)) {

FamilyBudget family = method

.getAnnotation(FamilyBudget.class);

String userRole = family.userRole();

int budgetLimit = family.budgetLimit();

if (userRole.equals(role)) {

if(spend<=budgetLimit){

method.invoke(FamilyMember.class.newInstance(),

budgetLimit, spend);

}else{

System.out.println("Budget Limit Over");

}

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

testCases--;

}

}

}

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 <= N <= 10
* 0 <= MoneySpend <= 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.

#### **Sample Input**

3

SENIOR 75

JUNIOR 45

SENIOR 40

#### **Sample Output**

Senior Member

Spend: 75

Budget Left: 25

Junior Member

Spend: 45

Budget Left: 5

Senior Member

Spend: 40

Budget Left: 60

## ****Solution – Java Annotations****

import java.lang.annotation.\*;

import java.lang.reflect.\*;

import java.util.\*;

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@interface FamilyBudget {

String userRole() default "GUEST";

int budgetLimit() default 0;

}

class FamilyMember {

@FamilyBudget(userRole = "SENIOR", budgetLimit = 100)

public void seniorMember(int budget, int moneySpend) {

System.out.println("Senior Member");

System.out.println("Spend: " + moneySpend);

System.out.println("Budget Left: " + (budget - moneySpend));

}

@FamilyBudget(userRole = "JUNIOR", budgetLimit = 50)

public void juniorUser(int budget, int moneySpend) {

System.out.println("Junior Member");

System.out.println("Spend: " + moneySpend);

System.out.println("Budget Left: " + (budget - moneySpend));

}

}

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int testCases = Integer.parseInt(in.nextLine());

while (testCases > 0) {

String role = in.next();

int spend = in.nextInt();

try {

Class annotatedClass = FamilyMember.class;

Method[] methods = annotatedClass.getMethods();

for (Method method : methods) {

if (method.isAnnotationPresent(FamilyBudget.class)) {

FamilyBudget family = method

.getAnnotation(FamilyBudget.class);

String userRole = family.userRole();

int budgetLimit = family.budgetLimit();

if (userRole.equals(role)) {

if(spend<=budgetLimit){

method.invoke(FamilyMember.class.newInstance(),

budgetLimit, spend);

}else{

System.out.println("Budget Limit Over");

}

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

testCases--;

}

}

}

# Covariant Return Types | HackerRank Solution

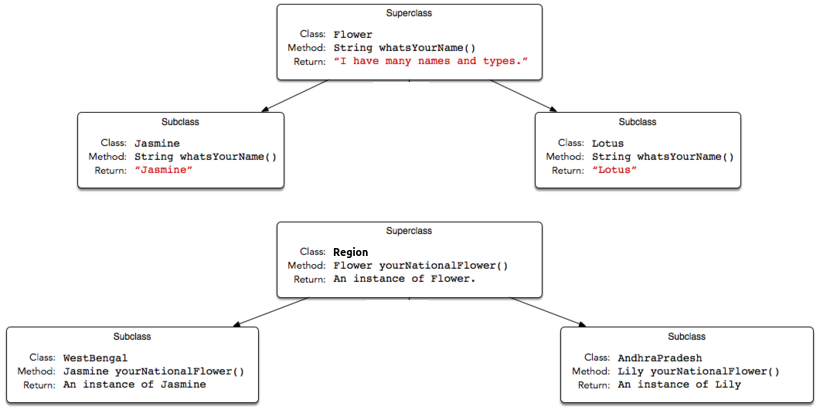
Hello coders, today we are going to solve **Covariant Return Types**[**HackerRank**](https://www.codingbroz.com/covariant-return-types-solution/)**Solution in Java**.

## ****Problem****

Java allows for Covariant Return Types, which means you can vary your return type as long you are returning a subclass of your specified return type.

Method Overriding allows a subclass to override the behavior of an existing superclass method and specify a return type that is some subclass of the original return type. It is best practice to use the @Override annotation when overriding a superclass method.

Implement the classes and methods detailed in the diagram below:



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.

## ****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**

AndhraPradesh

#### **Sample Input**

Lily

#### **Explanation**

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.

## ****Solution – Covariant Return Types in Java****

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

//Complete the classes below

class Flower{

String whatsYourName(){

return "I have many names and types.";

}

}

class Jasmine extends Flower{

@Override

String whatsYourName(){

return "Jasmine";

}

}

class Lily extends Flower{

@Override

String whatsYourName(){

return "Lily";

}

}

class Lotus extends Flower{

@Override

String whatsYourName(){

return "Lotus";

}

}

class Region{

Flower yourNationalFlower(){

return new Flower();

}

}

class WestBengal extends Region{

@Override

Jasmine yourNationalFlower(){

return new Jasmine();

}

}

class Karnataka extends Region{

@Override

Lotus yourNationalFlower(){

return new Lotus();

}

}

class AndhraPradesh extends Region{

@Override

Lily yourNationalFlower(){

return new Lily();

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String s = reader.readLine().trim();

Region region = null;

switch (s) {

case "WestBengal":

region = new WestBengal();

break;

case "AndhraPradesh":

region = new AndhraPradesh();

break;

}

Flower flower = region.yourNationalFlower();

System.out.println(flower.whatsYourName());

}

}

# Java Lambda Expressions | HackerRank Solution

[Leave a Comment](https://www.codingbroz.com/java-lambda-expressions-solution/#respond) / [HackerRank](https://www.codingbroz.com/category/hackerrank/), [HackerRank Java](https://www.codingbroz.com/category/hackerrank/hackerrank-java/) / By [CodeBros](https://www.codingbroz.com/author/codingbroz0718/" \o "View all posts by CodeBros)

Hello coders, today we are going to solve **Java Lambda Expressions**[**HackerRank**](https://www.codingbroz.com/java-lambda-expressions-solution/)**Solution**.

## ****Problem****

This Java 8 challenge tests your knowledge of Lambda expressions!

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

* PerformOperation isOdd(): The lambda expression must return **true**if a number is odd or **false**if it is even.
* PerformOperation isPrime(): The lambda expression must return **true** if a number is prime or **false** if it is composite.
* 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

## ****Solution – Java Lambda Expressions****

import java.io.\*;

import java.util.\*;

interface PerformOperation {

boolean check(int a);

}

class MyMath {

public static boolean checker(PerformOperation p, int num) {

return p.check(num);

}

// Write your code here

public PerformOperation isOdd() {

return (a) -> {

return (a % 2 == 1);

};

}

public PerformOperation isPrime() {

return (a) -> {

for (int i = 2; i <= a / 2; i++) {

if (a % i == 0)

return false;

}

return true;

};

}

public PerformOperation isPalindrome() {

return (a) -> {

int rev = 0;int r = 0;int n = a;

while (n != 0) {

r = n % 10;

rev = rev \* 10 + r;

n /= 10;

}

return (rev == a);

};

}

}

public class Solution {

public static void main(String[] args) throws IOException {

MyMath ob = new MyMath();

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int T = Integer.parseInt(br.readLine());

PerformOperation op;

boolean ret = false;

String ans = null;

while (T--> 0) {

String s = br.readLine().trim();

StringTokenizer st = new StringTokenizer(s);

int ch = Integer.parseInt(st.nextToken());

int num = Integer.parseInt(st.nextToken());

if (ch == 1) {

op = ob.isOdd();

ret = ob.checker(op, num);

ans = (ret) ? "ODD" : "EVEN";

} else if (ch == 2) {

op = ob.isPrime();

ret = ob.checker(op, num);

ans = (ret) ? "PRIME" : "COMPOSITE";

} else if (ch == 3) {

op = ob.isPalindrome();

ret = ob.checker(op, num);

ans = (ret) ? "PALINDROME" : "NOT PALINDROME";

}

System.out.println(ans);

}

}

}

# Java MD5 | HackerRank Solution

Hello coders, today we are going to solve **Java MD5**[**HackerRank**](https://www.codingbroz.com/java-md5-solution/)**Solution**.

## ****Problem****

MD5 (Message-Digest algorithm 5) is a widely-used [cryptographic](https://www.codingbroz.com/java-md5-solution/) hash function with a -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 <= |s| <= 20**
* String **s**consists of English alphabetic letters (i.e.,**[a-z A-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

## ****Solution – Java MD5****

import java.util.Scanner;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class Solution {

public static void main(String[] args) throws NoSuchAlgorithmException {

/\* Read and save the input String \*/

Scanner scan = new Scanner(System.in);

String str = scan.next();

scan.close();

/\* Encode the String using MD5 \*/

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(str.getBytes());

byte[] digest = md.digest();

/\* Print the encoded value in hexadecimal \*/

for (byte b : digest) {

System.out.format("%02x", b);

}

}

}

# Java SHA-256 | hackerRank Solution

Hello coders, today we are going to solve **Java SHA-256**[**HackerRank**](https://www.codingbroz.com/java-sha-256-solution/)**Solution**.

## ****Problem****

Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed hash (i.e., the output produced by executing a hashing algorithm) to a known and expected hash value, a person can determine the data’s integrity. For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can show whether the download has been modified or tampered with. In addition, cryptographic hash functions are extremely collision-resistant; in other words, it should be extremely difficult to produce the same hash output from two different input values using a 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 <= |s| <= 20**
* String **s**consists of English alphabetic letters (i.e.,**[a-z A-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**

68e109f0f40ca72a15e05cc22786f8e6

#### **Sample Input 1**

Javarmi123

#### **Sample Output 1**

2da2d1e0ce7b4951a858ed2d547ef485

## ****Solution – Java SHA-256****

import java.util.\*;

import java.security.\*;

public class Solution {

public static void main(String[] args) throws NoSuchAlgorithmException {

Scanner input = new Scanner(System.in);

MessageDigest m = MessageDigest.getInstance("SHA-256");

m.reset();

m.update(input.nextLine().getBytes());

for (byte i : m.digest()) {

System.out.print(String.format("%02x", i));

}

System.out.println();

}

}

//Question: tree leaf

class SumInLeavesVisitor extends TreeVis {

private int sumTreeLeaves = 0;

public int getResult() {

return sumTreeLeaves;

}

public void visitNode(TreeNode node) {

}

public void visitLeaf(TreeLeaf leaf) {

sumTreeLeaves += leaf.getValue();

}

}

class ProductOfRedNodesVisitor extends TreeVis {

private BigInteger result = BigInteger.ONE;

public int getResult() {

return (result.mod(BigInteger.valueOf(1000000007))).intValue();

}

public void visitNode(TreeNode node) {

computeResult(node);

}

public void visitLeaf(TreeLeaf leaf) {

computeResult(leaf);

}

private void computeResult(Tree tree) {

if (tree.getColor() == Color.RED) {

BigInteger value = tree.getValue() == 0 ? BigInteger.ONE : BigInteger.valueOf(tree.getValue());

result = result.multiply(value);

}

}

}

class FancyVisitor extends TreeVis {

private int sumEvenTreeNodes = 0;

private int sumGreenTreeLeaves = 0;

public int getResult() {

int difference = sumEvenTreeNodes - sumGreenTreeLeaves;

return difference > 0 ? difference : -difference;

}

public void visitNode(TreeNode node) {

if (node.getDepth() % 2 == 0) {

sumEvenTreeNodes += node.getValue();

}

}

public void visitLeaf(TreeLeaf leaf) {

if (leaf.getColor() == Color.GREEN) {

sumGreenTreeLeaves += leaf.getValue();

}

}

}

class Edge implements Comparable<Edge>{

public final int vertex1;

public final int vertex2;

public Edge(int vertex1, int vertex2) {

this.vertex1 = vertex1;

this.vertex2 = vertex2;

}

@Override

public int compareTo(Edge edge) {

if (vertex1 != edge.vertex1) {

return vertex1 - edge.vertex1;

}else{

return vertex2 - edge.vertex2;

}

}

}

public class Solution {

public static Tree solve() {

try (Scanner scanner = new Scanner(System.in);)

{

int n = scanner.nextInt();

int[] listValues = new int[n];

Color[] listColors = new Color[n];

Edge[] edges = new Edge[n];

int[] numberOfReferences = new int[n];

Tree[] listTrees = new Tree[n];

//Scanning values

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

listValues[i] = scanner.nextInt();

}

//Scanning colors

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

listColors[i] = scanner.nextInt() == 0 ? Color.RED : Color.GREEN;

}

//Scanning edges and counting number of references on a tree

for (int i = 1; i < n; i++) {

int vertex1 = scanner.nextInt() - 1;

int vertex2 = scanner.nextInt() - 1;

edges[i] = new Edge(vertex1, vertex2);

numberOfReferences[vertex1]++;

numberOfReferences[vertex2]++;

}

//Building trees

listTrees[0] = new TreeNode(listValues[0], listColors[0], 0);

for (int i = 1; i < n; i++) {

//Finding an edge referencing an already existing vertex

for

(int j = i + 1; listTrees[edges[i].vertex1] == null && listTrees[edges[i].vertex2] == null; j++) {

Edge tempEdge = edges[i];

edges[i] = edges[j];

edges[j] = tempEdge;

}

//Adding the tree

int parent = listTrees[edges[i].vertex1] != null ? edges[i].vertex1 : edges[i].vertex2;

int child = listTrees[edges[i].vertex1] == null ? edges[i].vertex1 : edges[i].vertex2;

TreeNode parentNode = (TreeNode)(listTrees[parent]);

Tree childTree = null;

if (numberOfReferences[child] > 1) {

childTree = new TreeNode(

listValues[child],

listColors[child],

parentNode.getDepth() + 1

);

}else{

childTree = new TreeLeaf(

listValues[child],

listColors[child],

parentNode.getDepth() + 1

);

}

listTrees[child] = childTree;

parentNode.addChild(childTree);

}

return listTrees[0];

}

}

//better tree leaf

class SumInLeavesVisitor extends TreeVis {

long sum = 0;

public int getResult() {

return (int) sum;

}

public void visitNode(TreeNode node) {

}

public void visitLeaf(TreeLeaf leaf) {

sum += leaf.getValue();

}

}

class ProductOfRedNodesVisitor extends TreeVis {

BigInteger prod = BigInteger.ONE;

public int getResult() {

return prod.mod(BigInteger.valueOf(1000000007)).intValue();

}

public void visitNode(TreeNode node) {

if (node.getColor() == Color.RED) {

prod = prod.multiply(BigInteger.valueOf(node.getValue()));

}

}

public void visitLeaf(TreeLeaf leaf) {

if (leaf.getColor() == Color.RED) {

prod = prod.multiply(BigInteger.valueOf(leaf.getValue()));

}

}

}

class FancyVisitor extends TreeVis {

long evenSum = 0;

long greenSum = 0;

public int getResult() {

return (int) Math.abs(evenSum - greenSum);

}

public void visitNode(TreeNode node) {

if (node.getDepth() % 2 == 0) {

evenSum += node.getValue();

}

}

public void visitLeaf(TreeLeaf leaf) {

if (leaf.getColor() == Color.GREEN) {

greenSum += leaf.getValue();

}

}

}

public class Solution {

public static Tree solve() {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

ns = new Node[n + 1]; // ignore 0

for (int i = 1; i < n + 1; i++) {

ns[i] = new Node();

ns[i].val = in.nextInt();

}

for (int i = 1; i < n + 1; i++) {

ns[i].col = in.nextInt() == 1 ? Color.GREEN : Color.RED;

}

for (int i = 0; i < n - 1; i++) {

int from = in.nextInt();

int to = in.nextInt();

ns[from].connections.add(to);

ns[to].connections.add(from);

}

Tree ret = buildNode(1, 0, 0);

return ret;

}

static Tree buildNode(int idx, int depth, int parent) {

Node n = ns[idx];

if (n.connections.contains(parent)) {

n.connections.remove(parent);

}

if (n.connections.size() != 0) {

TreeNode ret = new TreeNode(n.val, n.col, depth);

for (int i : n.connections) {

ret.addChild(buildNode(i, depth + 1, idx));

}

return ret;

} else {

return new TreeLeaf(n.val, n.col, depth);

}

}

static Node[] ns;

static class Node {

int val;

Color col;

HashSet<Integer> connections = new HashSet<>();

}

# Q: Array Manipulation

Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in the array.

**Example**

Queries are interpreted as follows:

a b k

1 5 3

4 8 7

6 9 1

Add the values of  between the indices  and  inclusive:

index-> 1 2 3 4 5 6 7 8 9 10

[0,0,0, 0, 0,0,0,0,0, 0]

[3,3,3, 3, 3,0,0,0,0, 0]

[3,3,3,10,10,7,7,7,0, 0]

[3,3,3,10,10,8,8,8,1, 0]

The largest value is  after all operations are performed.

**Function Description**

Complete the function *arrayManipulation* in the editor below.

arrayManipulation has the following parameters:

* *int n* - the number of elements in the array
* *int queries[q][3]* - a two dimensional array of queries where each *queries[i]* contains three integers, *a*, *b*, and *k*.

**Returns**

* *int* - the maximum value in the resultant array

**Input Format**

The first line contains two space-separated integers  and , the size of the array and the number of operations.  
Each of the next  lines contains three space-separated integers ,  and , the left index, right index and summand.

**Constraints**

**Sample Input**

5 3

1 2 100

2 5 100

3 4 100

**Sample Output**

200

**Explanation**

After the first update the list is 100 100 0 0 0.  
After the second update list is 100 200 100 100 100.  
After the third update list is 100 200 200 200 100.

The maximum value is 200.





import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.Scanner;

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(new BufferedReader(new InputStreamReader(System.in), 256 << 10));

int n = in.nextInt();

long[] delta = new long[n+1];

int m = in.nextInt();

for (int q = 0; q < m; q++) {

int l = in.nextInt();

int r = in.nextInt();

int k = in.nextInt();

delta[l-1] += k;

delta[r] -= k;

}

long max = 0;

long c = 0;

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

c += delta[i];

if (c > max) max = c;

}

System.out.println(max);

}

}

Others:

long[] baseArr = new long[n+1];

long maxSum = 0;

if(queries == null || queries.size() ==0){

return 0;

}

for(List<Integer> query : queries){

int start = query.get(0)-1;

int end = query.get(1) - 1;

baseArr[start]+= query.get(2);

baseArr[end+1]-= query.get(2);

}

int sum=0;

for(long item : baseArr){

sum+=item;

maxSum = Math.max(sum,maxSum);

}

return maxSum;

}

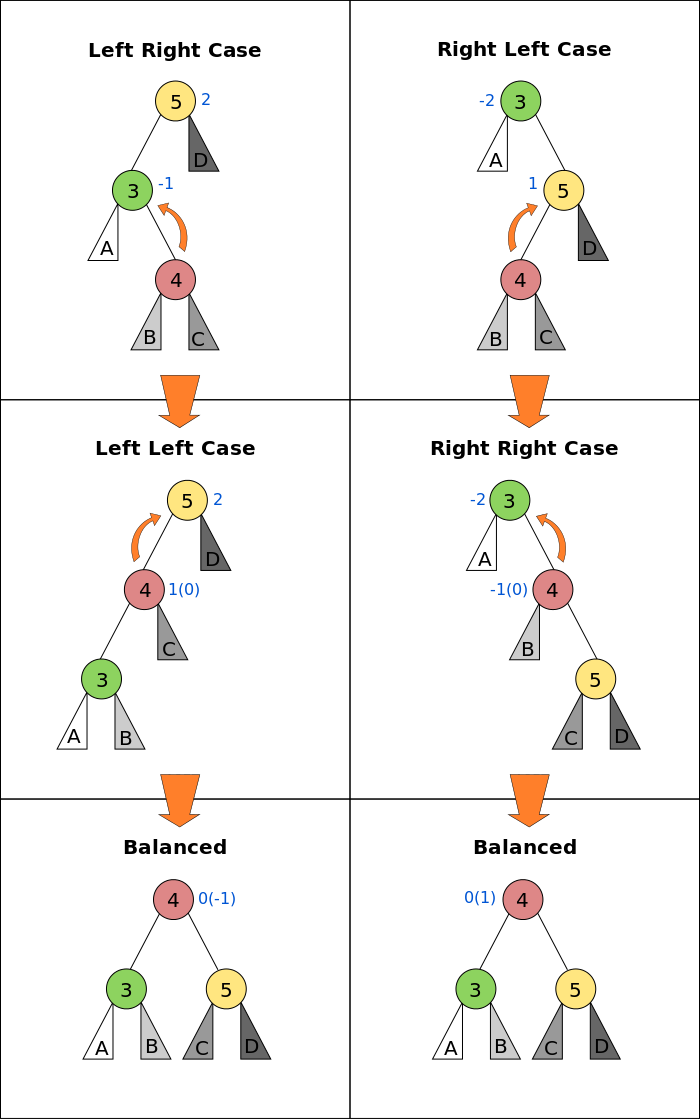
# Self Balancing Tree

An AVL tree (Georgy Adelson-Velsky and Landis' tree, named after the inventors) is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

We define balance factor for each node as :

balanceFactor = height(left subtree) - height(right subtree)

The balance factor of any node of an AVL tree is in the integer range [-1,+1]. If after any modification in the tree, the balance factor becomes less than −1 or greater than +1, the subtree rooted at this node is unbalanced, and a rotation is needed.



(<https://en.wikipedia.org/wiki/AVL_tree>)

You are given a pointer to the root of an AVL tree. You need to insert a value into this tree and perform the necessary rotations to ensure that it remains balanced.

**Input Format**

You are given a function,

node \*insert(node \* root,int new\_val)

{

}

'node' is defined as :

struct node

{

int val; //value

struct node\* left; //left child

struct node\* right; //right child

int ht; //height of the node

} node;

You only need to complete the function.

Note: All the values in the tree will be distinct. Height of a Null node is -1 and the height of the leaf node is 0.

**Output Format**

Insert the new value into the tree and return a pointer to the root of the tree. Ensure that the tree remains balanced.

**Sample Input**

3

/ \

2 4

\

5

The value to be inserted is 6.

**Sample Output**

3

/ \

2 5

/ \

4 6

**Explanation**

After inserting 6 in the tree. the tree becomes:

3 (Balance Factor = -2)

/ \

2 4 (Balance Factor = -2)

\

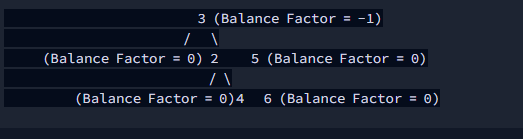
5 (Balance Factor = -1)

\

6 (Balance Factor = 0)

Balance Factor of nodes 3 and 4 is no longer in the range [-1,1]. We need to perform a rotation to balance the tree. This is the right right case. We perform a single rotation to balance the tree.

After performing the rotation, the tree becomes :

 3 (Balance Factor = -1)

/ \

(Balance Factor = 0) 2 5 (Balance Factor = 0)

/ \

(Balance Factor = 0)4 6 (Balance Factor = 0)

/\* Class node is defined as :

class Node

int val; //Value

int ht; //Height

Node left; //Left child

Node right; //Right child

\*/

static int h(Node x)

{

return x != null ? x.ht : -1;

}

static void updateh(Node x)

{

if(x == null)return;

x.ht = Math.max(h(x.left), h(x.right)) + 1;

}

static Node insert(Node root,int val)

{

if(root == null){

Node n = new Node();

n.val = val;

n.ht = 0;

return n;

}

if(val < root.val){

root.left = insert(root.left, val);

int bf = h(root.left) - h(root.right);

if(bf > 1){

Node A = root.left;

int bfl = h(root.left.left) - h(root.left.right);

if(bfl < 0){

Node B = root.left.right;

root.left = B;

A.right = B.left;

B.left = A;

updateh(A);

updateh(B);

}

}

if(bf > 1){

Node A = root.left;

Node B = A.left;

root.left = A.right;

A.right = root;

updateh(root);

updateh(A);

root = A;

}

}else{

root.right = insert(root.right, val);

int bf = h(root.left) - h(root.right);

if(bf < -1){

Node A = root.right;

int bfl = h(root.right.left) - h(root.right.right);

if(bfl > 0){

Node B = root.right.left;

root.right = B;

A.left = B.right;

B.right = A;

updateh(A);

updateh(B);

}

}

if(bf < -1){

Node A = root.right;

Node B = A.right;

root.right = A.left;

A.left = root;

updateh(root);

updateh(A);

root = A;

}

}

updateh(root);

return root;

}

Question:

1. [**Prepare**](https://www.hackerrank.com/dashboard)
2. [Data Structures](https://www.hackerrank.com/domains/data-structures)
3. [Balanced Trees](https://www.hackerrank.com/domains/data-structures/balanced-trees)
4. Array and simple queries

Given two numbers NNN and MMM. NNN indicates the number of elements in the array A[]A[ ]A[] (1−indexed)(1 - indexed)(1−indexed) and MMM indicates number of queries.

You need to perform two types of queries on the array A[]A[ ]A[].

You are given MMM queries. Queries can be of two types, type 1 and type 2.

* Type 1 queries are represented as 1  i  j1 \; i \; j1ij : Modify the given array by removing elements from iii to jjj and adding them to the front.
* Type 2 queries are represented as 2  i  j2 \; i \; j2ij : Modify the given array by removing elements from iii to jjj and adding them to the back.

Your task is to simply print ∣A[1]−A[N]∣|A[1] - A[N]|∣A[1]−A[N]∣ of the resulting array after the execution of MMM queries followed by the resulting array.

**Note**: While adding at back or front the order of elements is preserved.

**Input Format**

First line consists of two space-separated integers,  and .  
Second line contains  integers, which represent the elements of the array.  
 queries follow. Each line contains a query of either *type 1* or *type 2* in the form

**Constraints**

**Output Format**

Print the absolute value i.e.  in the first line.  
Print elements of the resulting array in the second line. Each element should be seperated by a single space.

**Sample Input**

8 4

1 2 3 4 5 6 7 8

1 2 4

2 3 5

1 4 7

2 1 4

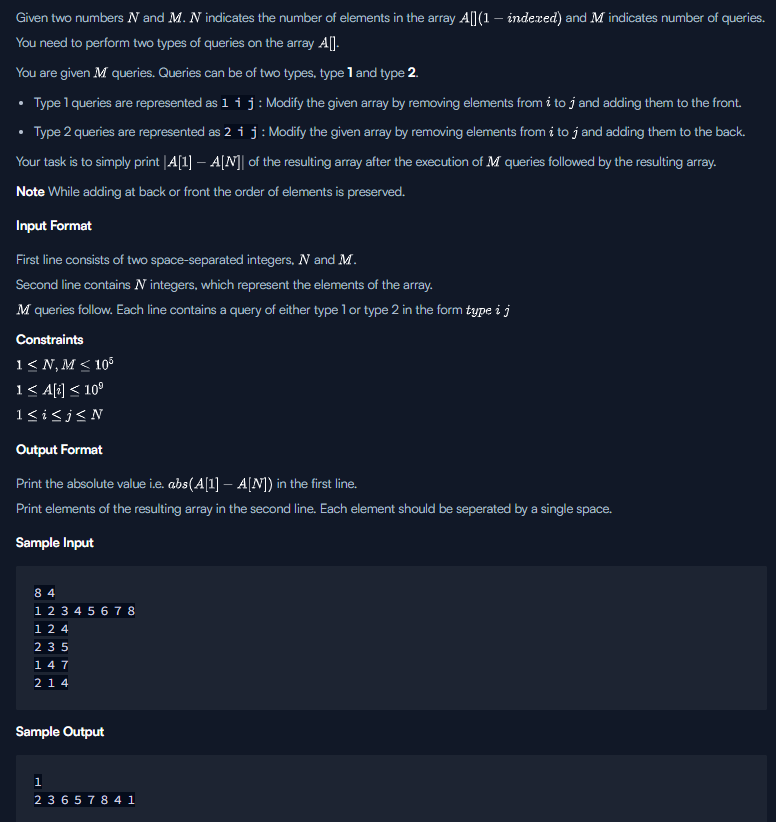
**Sample Output**

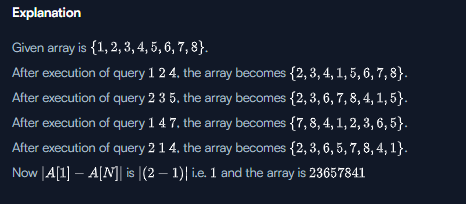
1

2 3 6 5 7 8 4 1

**Explanation**

Given array is .  
After execution of query , the array becomes .  
After execution of query , the array becomes .  
After execution of query , the array becomes .  
After execution of query , the array becomes .  
Now  is  i.e.  and the array is





//code with time out, but easy to understand:

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input the size of the array and the number of commands

int N = sc.nextInt();

int M = sc.nextInt();

// Input the initial elements of the array

List<Integer> array = new ArrayList<>();

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

array.add(sc.nextInt());

}

// Process each command

while (M-- > 0) {

int cmd = sc.nextInt();

int s = sc.nextInt() - 1; // Convert to 0-based index

int e = sc.nextInt(); // e is exclusive in subList

List<Integer> sublist = new ArrayList<>(array.subList(s, e));

array.subList(s, e).clear();

if (cmd == 1) {

array.addAll(0, sublist); // Add at the beginning

} else if (cmd == 2) {

array.addAll(sublist); // Add at the end

}

}

// Output the result

System.out.println(Math.abs(array.get(0) - array.get(array.size() - 1)));

for (int value : array) {

System.out.print(value + " ");

}

}

}

Solution:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

Node root;

public Solution() {

root = null;

}

// Get the size of the subtree rooted at the given node

public int size(Node root) {

if (root == null)

return 0;

return root.size;

}

// Update the size of the subtree rooted at the given node

private void update(Node root) {

if (root != null)

root.size = 1 + size(root.left) + size(root.right);

}

// Split the Treap into two subtrees such that all nodes in the left subtree have indices < key

// and all nodes in the right subtree have indices >= key

public DNode split(Node root, int key) {

DNode pair = new DNode();

if (root == null) {

return pair;

}

if (size(root.left) >= key) {

pair = split(root.left, key);

root.left = pair.right;

update(root);

pair.right = root;

} else {

pair = split(root.right, key - size(root.left) - 1);

root.right = pair.left;

update(root);

pair.left = root;

}

return pair;

}

// Merge two Treaps into one

public Node merge(Node left, Node right) {

if (left == null) return right;

if (right == null) return left;

if (left.pri > right.pri) {

left.right = merge(left.right, right);

update(left);

return left;

} else {

right.left = merge(left, right.left);

update(right);

return right;

}

}

// Get the first (smallest) node in the Treap

public Node getFirst() {

if (root == null)

return null;

return getFirst(root);

}

private Node getFirst(Node root) {

if (root.left != null)

return getFirst(root.left);

return root;

}

// Get the last (largest) node in the Treap

public Node getLast() {

if (root == null)

return null;

return getLast(root);

}

private Node getLast(Node root) {

if (root.right != null)

return getLast(root.right);

return root;

}

// Perform query type 1: Move the elements in the range to the front

public void query1(int left, int right) {

DNode pair1 = split(root, right);

DNode pair2 = split(pair1.left, left-1);

root = merge(pair2.left, pair1.right);

root = merge(pair2.right, root);

}

// Perform query type 2: Move the elements in the range to the back

public void query2(int left, int right) {

DNode pair1 = split(root, right);

DNode pair2 = split(pair1.left, left-1);

root = merge(pair2.left, pair1.right);

root = merge(root, pair2.right);

}

// Add a new node to the Treap

public void add(int val, int pri) {

Node n = new Node(val, pri);

root = merge(root, n);

}

// In-order traversal of the Treap

public void inorder() {

inorder(root);

System.out.println();

}

private void inorder(Node n) {

if (n == null)

return;

inorder(n.left);

System.out.print(n + " ");

inorder(n.right);

}

private static class DNode {

Node left;

Node right;

DNode() {

left = null;

right = null;

}

public String toString() {

return "L:" + left + " R:" + right;

}

}

private static class Node {

int val;

int pri;

int size;

Node left;

Node right;

Node(int val, int pri) {

this.val = val;

this.pri = pri;

size = 1;

left = null;

right = null;

}

public String toString() {

return String.valueOf(val);

}

}

public static void main(String[] args) {

Solution t = new Solution();

Scanner sc = new Scanner(System.in);

Random rd = new Random();

int n = sc.nextInt();

int m = sc.nextInt();

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

int v = sc.nextInt();

t.add(v, rd.nextInt(n \* 10));

}

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

int op = sc.nextInt();

int low = sc.nextInt();

int high = sc.nextInt();

if (op == 1)

t.query1(low, high);

else

t.query2(low, high);

}

System.out.println(Math.abs(t.getFirst().val - t.getLast().val));

t.inorder();

sc.close();

}

}

Solution with color:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    Node root;

    public Solution() {

        root = null;

    }

    // Get the size of the subtree rooted at the given node

    public int size(Node root) {

        if (root == null)

            return 0;

        return root.size;

    }

    // Update the size of the subtree rooted at the given node

    private void update(Node root) {

        if (root != null)

            root.size = 1 + size(root.left) + size(root.right);

    }

    // Split the Treap into two subtrees such that all nodes in the left subtree have indices < key

    // and all nodes in the right subtree have indices >= key

    public DNode split(Node root, int key) {

        DNode pair = new DNode();

        if (root == null) {

            return pair;

        }

        if (size(root.left) >= key) {

            pair = split(root.left, key);

            root.left = pair.right;

            update(root);

            pair.right = root;

        } else {

            pair = split(root.right, key - size(root.left) - 1);

            root.right = pair.left;

            update(root);

            pair.left = root;

        }

        return pair;

    }

    // Merge two Treaps into one

    public Node merge(Node left, Node right) {

        if (left == null) return right;

        if (right == null) return left;

        if (left.pri > right.pri) {

            left.right = merge(left.right, right);

            update(left);

            return left;

        } else {

            right.left = merge(left, right.left);

            update(right);

            return right;

        }

    }

    // Get the first (smallest) node in the Treap

    public Node getFirst() {

        if (root == null)

            return null;

        return getFirst(root);

    }

    private Node getFirst(Node root) {

        if (root.left != null)

            return getFirst(root.left);

        return root;

    }

    // Get the last (largest) node in the Treap

    public Node getLast() {

        if (root == null)

            return null;

        return getLast(root);

    }

    private Node getLast(Node root) {

        if (root.right != null)

            return getLast(root.right);

        return root;

    }

    // Perform query type 1: Move the elements in the range to the front

    public void query1(int left, int right) {

        DNode pair1 = split(root, right);

        DNode pair2 = split(pair1.left, left-1);

        root = merge(pair2.left, pair1.right);

        root = merge(pair2.right, root);

    }

    // Perform query type 2: Move the elements in the range to the back

    public void query2(int left, int right) {

        DNode pair1 = split(root, right);

        DNode pair2 = split(pair1.left, left-1);

        root = merge(pair2.left, pair1.right);

        root = merge(root, pair2.right);

    }

    // Add a new node to the Treap

    public void add(int val, int pri) {

        Node n = new Node(val, pri);

        root = merge(root, n);

    }

    // In-order traversal of the Treap

    public void inorder() {

        inorder(root);

        System.out.println();

    }

    private void inorder(Node n) {

        if (n == null)

            return;

        inorder(n.left);

        System.out.print(n + " ");

        inorder(n.right);

    }

    private static class DNode {

        Node left;

        Node right;

        DNode() {

            left = null;

            right = null;

        }

        public String toString() {

            return "L:" + left + " R:" + right;

        }

    }

    private static class Node {

        int val;

        int pri;

        int size;

        Node left;

        Node right;

        Node(int val, int pri) {

            this.val = val;

            this.pri = pri;

            size = 1;

            left = null;

            right = null;

        }

        public String toString() {

            return String.valueOf(val);

        }

    }

    public static void main(String[] args) {

        Solution t = new Solution();

        Scanner sc = new Scanner(System.in);

        Random rd = new Random();

        int n = sc.nextInt();

        int m = sc.nextInt();

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

            int v = sc.nextInt();

            t.add(v, rd.nextInt(n \* 10));

        }

        for (int i = 0; i < m; i++) {

            int op = sc.nextInt();

            int low = sc.nextInt();

            int high = sc.nextInt();

            if (op == 1)

                t.query1(low, high);

            else

                t.query2(low, high);

        }

        System.out.println(Math.abs(t.getFirst().val - t.getLast().val));

        t.inorder();

        sc.close();

    }

}

Problem solving: pair

Given an array of integers and a target value, determine the number of pairs of array elements that have a difference equal to the target value.

**Example**

There are three values that differ by : , , and . Return .

**Function Description**

Complete the *pairs* function below.

pairs has the following parameter(s):

* *int k:* an integer, the target difference
* *int arr[n]:* an array of integers

**Returns**

* *int:* the number of pairs that satisfy the criterion

**Input Format**

The first line contains two space-separated integers  and , the size of  and the target value.  
The second line contains  space-separated integers of the array .

**Constraints**

* each integer  will be unique

**Sample Input**

STDIN Function

----- --------

5 2 arr[] size n = 5, k =2

1 5 3 4 2 arr = [1, 5, 3, 4, 2]

**Sample Output**

3

**Explanation**

There are 3 pairs of integers in the set with a difference of 2: [5,3], [4,2] and [3,1]. .



import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'pairs' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts following parameters:

\* 1. INTEGER k

\* 2. INTEGER\_ARRAY arr

\*/

public static int pairs(int k, List<Integer> arr) {

// Write your code here

HashSet<Integer> hs = new HashSet<>();

for(Integer i : arr)

hs.add(i);

int cnt=0;

for(Integer i : hs)

if(hs.contains(i+k))

cnt++;

return cnt;

// Collections.sort(arr, Collections.reverseOrder());

// int cnt=0;

// for(int i=0; i<arr.size(); i++){

// for(int j=i+1; j<arr.size();j++)

// {

// if(arr.get(i))

// }

// }

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

int n = Integer.parseInt(firstMultipleInput[0]);

int k = Integer.parseInt(firstMultipleInput[1]);

List<Integer> arr = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Integer::parseInt)

.collect(toList());

int result = Result.pairs(k, arr);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

# Sherlock and Array

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right.

**Example**

 is between two subarrays that sum to .

The answer is  since left and right sum to .

You will be given arrays of integers and must determine whether there is an element that meets the criterion. If there is, return YES. Otherwise, return NO.

**Function Description**

Complete the *balancedSums* function in the editor below.

balancedSums has the following parameter(s):

* *int arr[n]:* an array of integers

**Returns**

* *string:* either YES or NO

**Input Format**

The first line contains , the number of test cases.

The next  pairs of lines each represent a test case.  
- The first line contains , the number of elements in the array .  
- The second line contains  space-separated integers  where .

**Constraints**

**Sample Input 0**

2

3

1 2 3

4

1 2 3 3

**Sample Output 0**

NO

YES

**Explanation 0**

For the first test case, no such index exists.  
For the second test case, , therefore index  satisfies the given conditions.

**Sample Input 1**

3

5

1 1 4 1 1

4

2 0 0 0

4

0 0 2 0

**Sample Output 1**

YES

YES

YES

**Explanation 1**

In the first test case,  is between two subarrays summing to .  
In the second case,  is between two subarrays summing to .  
In the third case,  is between two subarrays summing to .





Solution:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'balancedSums' function below.

\*

\* The function is expected to return a STRING.

\* The function accepts INTEGER\_ARRAY arr as parameter.

\*/

public static String balancedSums(List<Integer> arr) {

// Write your code here

int rightsum=0;

int leftsum=0;

for(Integer i : arr)

{ rightsum+=i;}

for(Integer i : arr){

rightsum-=i;

if(rightsum==leftsum)

{ return "YES";}

leftsum+=i;

}

return "NO";

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int T = Integer.parseInt(bufferedReader.readLine().trim());

IntStream.range(0, T).forEach(TItr -> {

try {

int n = Integer.parseInt(bufferedReader.readLine().trim());

List<Integer> arr = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Integer::parseInt)

.collect(toList());

String result = Result.balancedSums(arr);

bufferedWriter.write(result);

bufferedWriter.newLine();

} catch (IOException ex) {

throw new RuntimeException(ex);

}

});

bufferedReader.close();

bufferedWriter.close();

}

}

# Maximum Subarray Sum

We define the following:

* A *subarray* of array  of length  is a contiguous segment from  through  where .
* The *sum* of an array is the sum of its elements.

Given an  element array of integers, , and an integer, , determine the maximum value of the sum of any of its subarrays modulo .

**Example**

The following table lists all subarrays and their moduli:

sum %2

[1] 1 1

[2] 2 0

[3] 3 1

[1,2] 3 1

[2,3] 5 1

[1,2,3] 6 0

The maximum modulus is .

**Function Description**

Complete the *maximumSum* function in the editor below.

maximumSum has the following parameter(s):

* *long a[n]:* the array to analyze
* *long m:* the modulo divisor

**Returns**  
- *long:* the maximum (subarray sum modulo )

**Input Format**

The first line contains an integer , the number of queries to perform.

The next  pairs of lines are as follows:

* The first line contains two space-separated integers  and (long), the length of  and the modulo divisor.
* The second line contains  space-separated long integers .

**Constraints**

* the sum of  over all test cases

**Sample Input**

STDIN Function

----- --------

1 q = 1

5 7 a[] size n = 5, m = 7

3 3 9 9 5

**Sample Output**

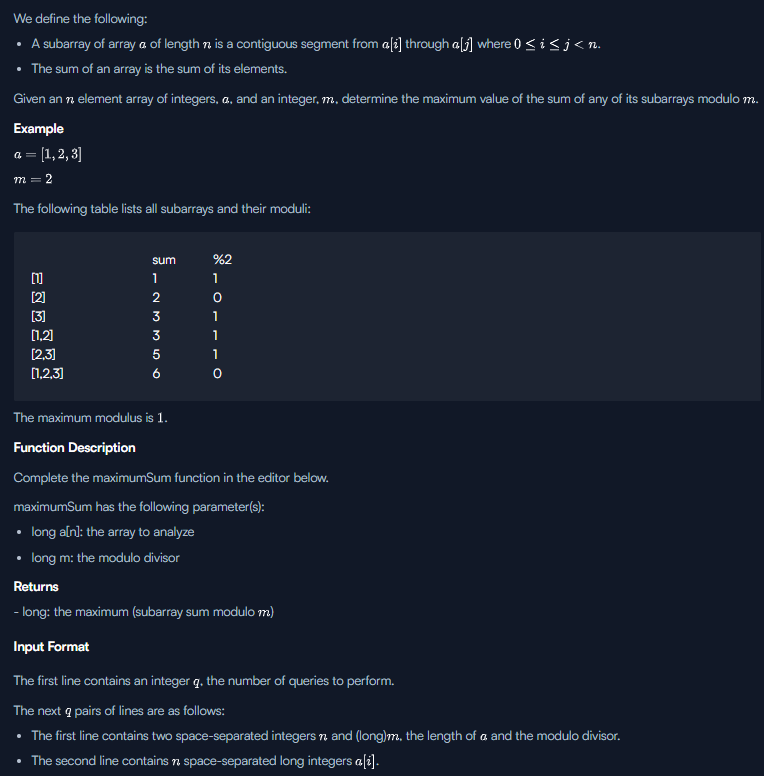
6

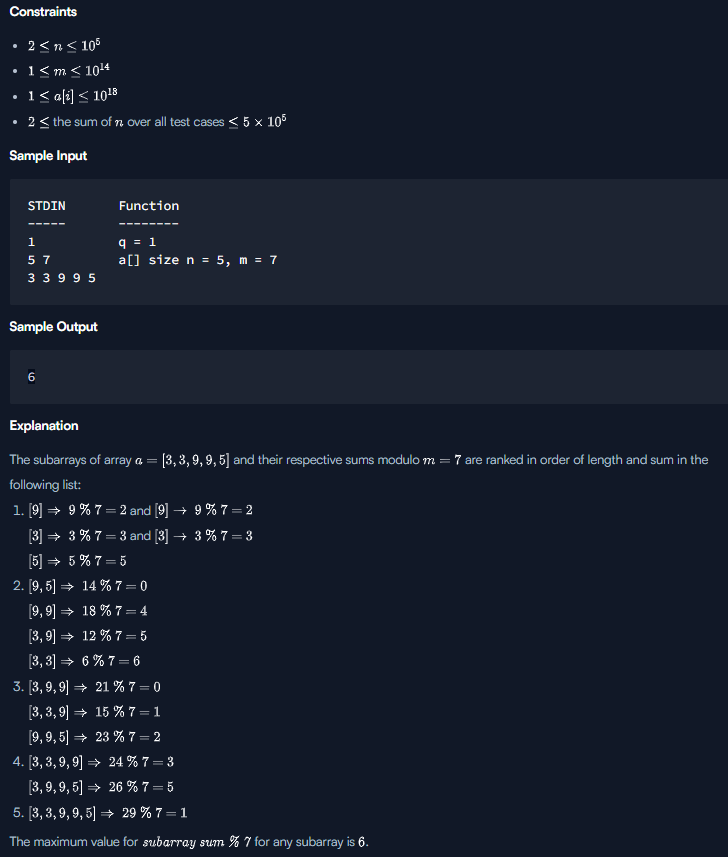
**Explanation**

The subarrays of array  and their respective sums modulo  are ranked in order of length and sum in the following list:

1. and   
    and

The maximum value for  for any subarray is .





Solution1:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

long T = scanner.nextLong();

int N = 0;

long M = 0;

long[] arr = new long[500000];

while (T-- > 0) {

N = scanner.nextInt(); // no of elements

M = scanner.nextLong(); // modulo by

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

arr[i] = scanner.nextLong();

}

long[] sumToIndex = new long[N];

sumToIndex[0] = arr[0] % M;

for (int i = 1; i < N; i++) {

sumToIndex[i] += (sumToIndex[i-1] + arr[i]) % M;

}

TreeMap<Long, Integer> treeMap = new TreeMap<Long, Integer>();

Long diff = Long.MIN\_VALUE;

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

if (!treeMap.containsKey(sumToIndex[i])) {

treeMap.put(sumToIndex[i], i);

Long higherKey = treeMap.higherKey(sumToIndex[i]);

if (higherKey != null) {

Long temp = M + (sumToIndex[i] - higherKey);

if (temp > diff) {

diff = temp;

}

}

}

}

long max = Long.MIN\_VALUE;

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

//System.out.println(sumToIndex[i]);

if (max < sumToIndex[i])

max = sumToIndex[i];

}

max = max > diff ? max : diff;

System.out.println(max);

}

}

}

Solution2:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

class Result {

/\*

\* Complete the 'maximumSum' function below.

\*

\* The function is expected to return a LONG\_INTEGER.

\* The function accepts following parameters:

\* 1. LONG\_INTEGER\_ARRAY a

\* 2. LONG\_INTEGER m

\*/

public static long maximumSum(List<Long> a, long m) {

// Write your code here

TreeSet<Long> sumSet = new TreeSet<Long>();

long best = 0;

long sum = 0;

for(int i = 0; i < a.size(); i++){

sum = (sum + a.get(i)) % m;

Long up = sumSet.higher(sum);

if(up == null){

best = Math.max(best,sum);

} else {

best = Math.max(best, m - up + sum);

}

sumSet.add(sum);

}

return best;

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int q = Integer.parseInt(bufferedReader.readLine().trim());

for (int qItr = 0; qItr < q; qItr++) {

String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

int n = Integer.parseInt(firstMultipleInput[0]);

long m = Long.parseLong(firstMultipleInput[1]);

String[] aTemp = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

List<Long> a = new ArrayList<>();

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

long aItem = Long.parseLong(aTemp[i]);

a.add(aItem);

}

long result = Result.maximumSum(a, m);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

}

bufferedReader.close();

bufferedWriter.close();

}

}

# Connected Cells in a Grid

Consider a matrix where each cell contains either a  or a . Any cell containing a  is called a *filled* cell. Two cells are said to be *connected* if they are adjacent to each other horizontally, vertically, or diagonally. In the following grid, all cells marked X are connected to the cell marked Y.

XXX

XYX

XXX

If one or more filled cells are also connected, they form a *region*. Note that each cell in a region is connected to zero or more cells in the region but is not necessarily directly connected to all the other cells in the region.

Given an  matrix, find and print the number of cells in the largest *region* in the matrix. Note that there may be more than one region in the matrix.

For example, there are two regions in the following  matrix. The larger region at the top left contains  cells. The smaller one at the bottom right contains .

110

100

001

**Function Description**

Complete the *connectedCell* function in the editor below.

connectedCell has the following parameter(s):  
- *int matrix[n][m]*:  represents the  row of the matrix

**Returns**  
- *int:* the area of the largest region

**Input Format**

The first line contains an integer , the number of rows in the matrix.  
The second line contains an integer , the number of columns in the matrix.  
Each of the next  lines contains  space-separated integers .

**Constraints**

**Sample Input**

STDIN Function

----- --------

4 n = 4

4 m = 4

1 1 0 0 grid = [[1, 1, 1, 0], [0, 1, 1, 0], [0, 0, 1, 0], [1, 0, 0, 0]]

0 1 1 0

0 0 1 0

1 0 0 0

**Sample Output**

5

**Explanation**

The diagram below depicts two regions of the matrix. Connected regions are filled with X or Y. Zeros are replaced with dots for clarity.

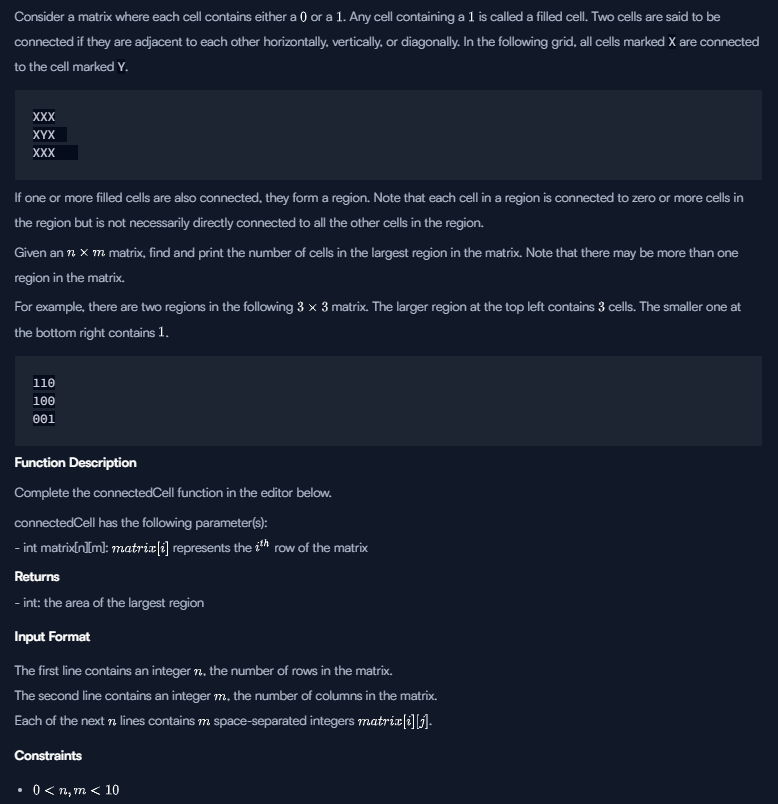
X X . .

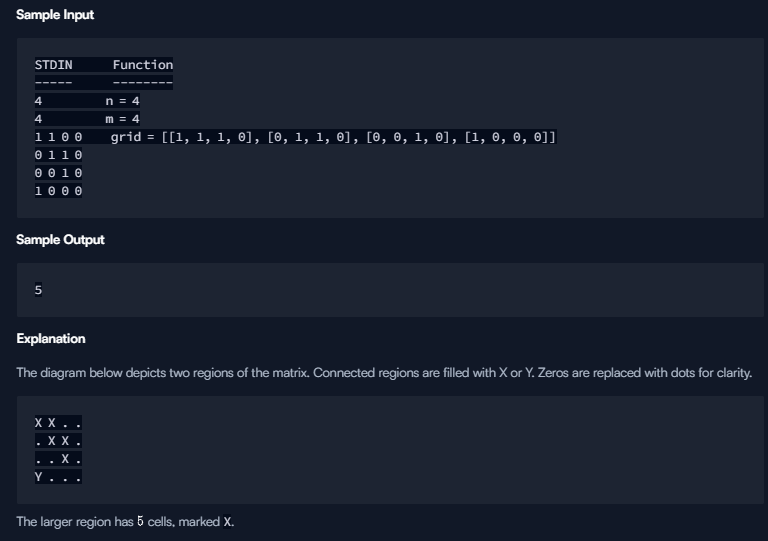
. X X .

. . X .

Y . . .

The larger region has  cells, marked X.





Solution: with run time error:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'connectedCell' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts 2D\_INTEGER\_ARRAY matrix as parameter.

\*/

static int countRegion(List<List<Integer>> matrix, int m, int n, int x, int y) {

if ((x < 0) || (x >= m) || (y < 0) || (y >= n) || (matrix.get(x).get(y) == 0))

return 0;

List<Integer> ma = matrix.get(x);

ma.set(y, 0);

matrix.set(x,ma);

return 1 + countRegion(matrix,m,n, x - 1, y)

+ countRegion(matrix,m,n,x + 1, y)

+ countRegion(matrix,m,n,x, y - 1)

+ countRegion(matrix,m,n,x, y + 1)

+ countRegion(matrix, m, n, x+1, y+1)

+ countRegion(matrix, m, n, x-1, y+1)

+ countRegion(matrix, m, n, x+1, y-1)

+ countRegion(matrix, m, n, x-1, y-1);

}

public static int connectedCell(List<List<Integer>> matrix) {

int maxRegion = 0;

int n = matrix.size();

int m = matrix.get(0).size();

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

for(int j=0; j<n; j++){

if(matrix.get(i).get(j) == 1){

int count = countRegion(matrix, m, n, i, j);

//System.out.println("count:"+count);

if(count > maxRegion)

maxRegion = count;

}

}

}

// System.out.println(maxRegion);

return maxRegion;

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int n = Integer.parseInt(bufferedReader.readLine().trim());

int m = Integer.parseInt(bufferedReader.readLine().trim());

List<List<Integer>> matrix = new ArrayList<>();

IntStream.range(0, n).forEach(i -> {

try {

matrix.add(

Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Integer::parseInt)

.collect(toList())

);

} catch (IOException ex) {

throw new RuntimeException(ex);

}

});

int result = Result.connectedCell(matrix);

bufferedWriter.write(String.valueOf(result));

bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}

Solution: no run time:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int m = in.nextInt();

int n = in.nextInt();

int[][] matrix = new int[m][n];

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

for(int j=0; j<n; j++){

matrix[i][j] = in.nextInt();

}

}

in.close();

int maxRegion = 0;

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

for(int j=0; j<n; j++){

if(matrix[i][j] == 1){

int count = countRegion(matrix, m, n, i, j);

//System.out.println("count:"+count);

if(count > maxRegion)

maxRegion = count;

}

}

}

System.out.println(maxRegion);

}

static int countRegion(int[][] matrix, int m, int n, int x, int y) {

if ((x < 0) || (x >= m) || (y < 0) || (y >= n) || (matrix[x][y] == 0))

return 0;

matrix[x][y] = 0;

return 1 + countRegion(matrix,m,n, x - 1, y)

+ countRegion(matrix,m,n,x + 1, y)

+ countRegion(matrix,m,n,x, y - 1)

+ countRegion(matrix,m,n,x, y + 1)

+ countRegion(matrix, m, n, x+1, y+1)

+ countRegion(matrix, m, n, x-1, y+1)

+ countRegion(matrix, m, n, x+1, y-1)

+ countRegion(matrix, m, n, x-1, y-1);

}

}

# Short Palindrome

Consider a string, , of  lowercase English letters where each character,  (, denotes the letter at index  in . We define an  palindromic tuple of  to be a sequence of indices in  satisfying the following criteria:

* , meaning the characters located at indices  and  are the same.
* , meaning the characters located at indices  and  are the same.
* , meaning that , , , and  are ascending in value and are valid indices within string .

Given , find and print the number of  tuples satisfying the above conditions. As this value can be quite large, print it modulo .

**Function Description**  
Complete the function shortPalindrome in the editor below.

shortPalindrome has the following paramter(s):  
- string s: a string

**Returns**  
- int: the number of tuples, modulo

**Input Format**

A single string, .

**Constraints**

* It is guaranteed that  only contains lowercase English letters.

**Sample Input 0**

kkkkkkz

**Sample Output 0**

15

**Explanation 0**

The letter z will not be part of a valid tuple because you need at least two of the same character to satisfy the conditions defined above. Because all tuples consisting of four k's are valid, we just need to find the number of ways that we can [choose](https://en.wikipedia.org/wiki/Binomial_coefficient) four of the six k's. This means our answer is .

**Sample Input 1**

ghhggh

**Sample Output 1**

4

**Explanation 1**

The valid tuples are:

Thus, our answer is .

**Sample Input 0**

kkkkkkz

**Sample Output 0**

15

**Sample Input 1**

abbaab

**Sample Output 1**

4

**Sample Input 2**

akakak

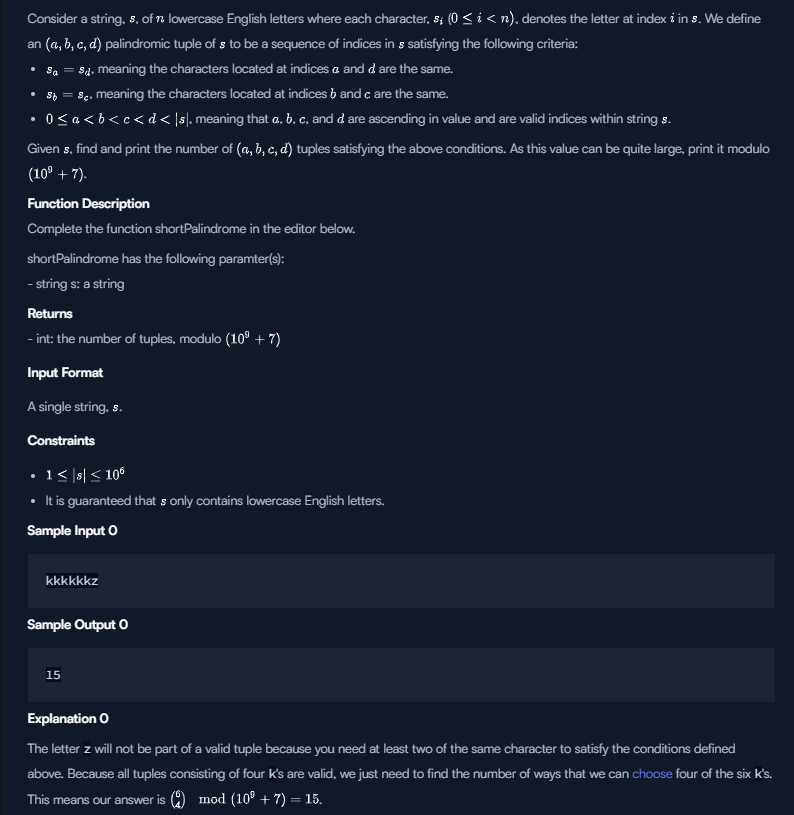
**Sample Output 2**

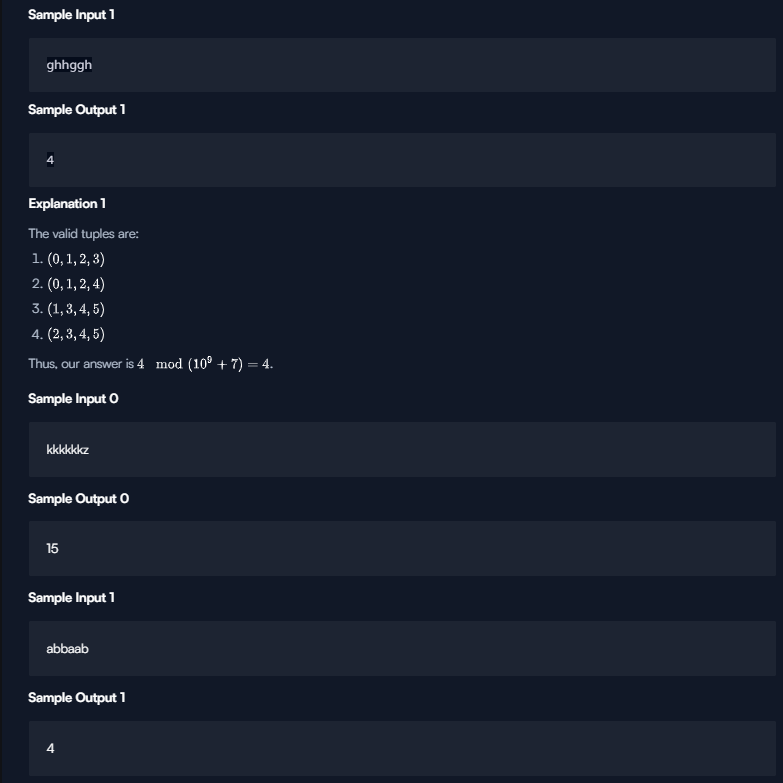
2

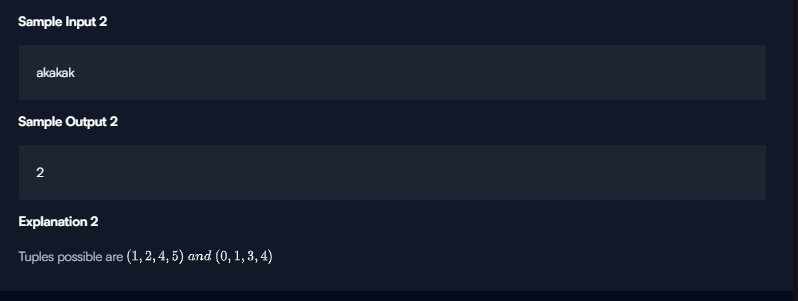
**Explanation 2**

Tuples possible are

Change Theme







Solution:

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Solution {

public static final int MOD = 1000000000 + 7;

public static void main(String[] args) {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String line = null;

try {

line = br.readLine();

br.close();

} catch (Exception ex) {

}

long[] suffixCount = new long[26];

long[][] suffixPairCount = new long[26][26];

for (int i = line.length() - 1; i >= 0; --i) {

int c = line.charAt(i) - 'a';

for (int j = 0; j < 26; j++) {

suffixPairCount[c][j] = (suffixPairCount[c][j] + suffixCount[j]) % MOD;

}

suffixCount[c] = (suffixCount[c] + 1) % MOD;

}

long[] prefixCount = new long[26];

long[][] prefixPairCount = new long[26][26];

long ret = 0;

for (int i = 0; i < line.length(); ++i) {

int c = line.charAt(i) - 'a';

for (int j = 0; j < 26; j++) {

long t = suffixCount[j];

if (j == c) t--;

t %= MOD;

ret = (ret + (prefixPairCount[j][c] \* t) % MOD) % MOD;

}

suffixCount[c] = (suffixCount[c] + MOD - 1) % MOD;

for (int j = 0; j < 26; j++) {

suffixPairCount[c][j] = (suffixPairCount[c][j] + MOD - suffixCount[j]) % MOD;

prefixPairCount[j][c] = (prefixPairCount[j][c] + prefixCount[j]) % MOD;

}

prefixCount[c] = (prefixCount[c] + 1) % MOD;

}

System.out.println(ret);

}

}

# Climbing the Leaderboard

An arcade game player wants to climb to the top of the leaderboard and track their ranking. The game uses [Dense Ranking](https://en.wikipedia.org/wiki/Ranking#Dense_ranking_.28.221223.22_ranking.29), so its leaderboard works like this:

* The player with the highest score is ranked number  on the leaderboard.
* Players who have equal scores receive the same ranking number, and the next player(s) receive the immediately following ranking number.

**Example**

The ranked players will have ranks , , , and , respectively. If the player's scores are ,  and , their rankings after each game are ,  and . Return .

**Function Description**

Complete the climbingLeaderboard function in the editor below.

climbingLeaderboard has the following parameter(s):

* int ranked[n]: the leaderboard scores
* int player[m]: the player's scores

**Returns**

* int[m]: the player's rank after each new score

**Input Format**

The first line contains an integer , the number of players on the leaderboard.  
The next line contains  space-separated integers , the leaderboard scores in decreasing order.  
The next line contains an integer, , the number games the player plays.  
The last line contains  space-separated integers , the game scores.

**Constraints**

* for
* for
* The existing leaderboard, , is in descending order.
* The player's scores, , are in ascending order.

**Subtask**

For  of the maximum score:

**Sample Input 1**

[Copy](https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem#!)[Download](https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem)

Array: ranked1001005040402010

Array: player52550120

7  
100 100 50 40 40 20 10  
4  
5 25 50 120

**Sample Output 1**

6  
4  
2  
1

**Explanation 1**

Alice starts playing with  players already on the leaderboard, which looks like this:

After Alice finishes game , her score is  and her ranking is :

After Alice finishes game , her score is  and her ranking is :

After Alice finishes game , her score is  and her ranking is tied with Caroline at :

After Alice finishes game , her score is  and her ranking is :

**Sample Input 2**

[Copy](https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem#!)[Download](https://www.hackerrank.com/challenges/climbing-the-leaderboard/problem)

Array: ranked1009090807560

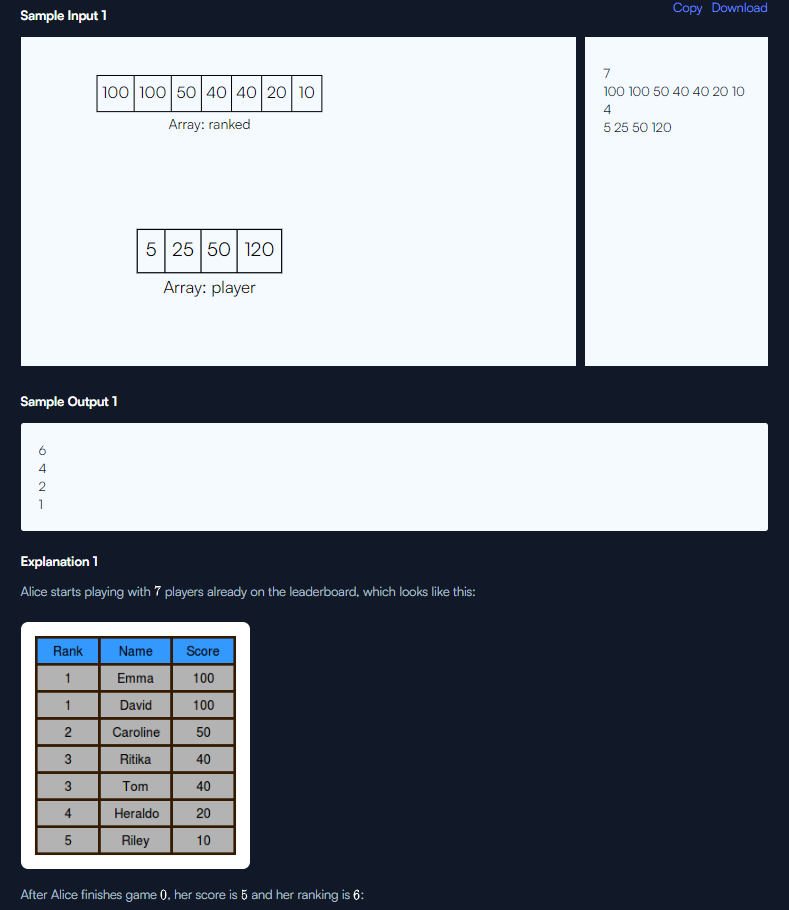
Array: player50657790102

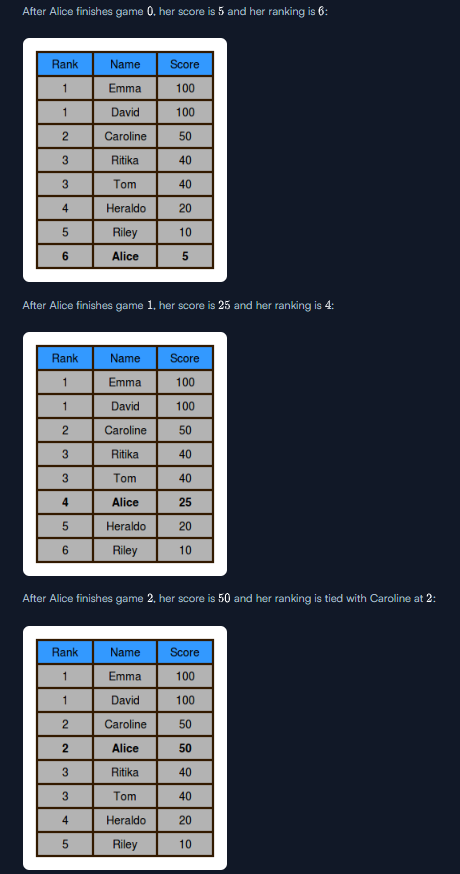
6  
100 90 90 80 75 60  
5  
50 65 77 90 102

**Sample Output 2**

6  
5  
4  
2  
1









Solution:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

List<Integer> scores = new ArrayList<Integer>();

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

int score = in.nextInt();

if (scores.size() == 0 || scores.get(scores.size() - 1) != score)

scores.add(score);

}

int m = in.nextInt();

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

int score = in.nextInt();

int min = 0;

int max = scores.size();

while (max > min){

int mid = (min + max) / 2;

if (scores.get(mid) <= score)

max = mid;

else

min = mid + 1;

}

System.out.println(min + 1);

}

}

}

Treemap:

TreeMap< String,Integer > ht=new TreeMap<>();

ht.put("12",1);

ht.put("22",2);

ht.put("32",3);

ht.put("42",4);

for(int i=0;i<ht.size();i++)

{

System.out.println(new Vector(ht.keySet()).get(i));

System.out.println(new Vector(ht.values()).get(i));

}

* [**Java Priority Queue – Hacker Rank Solution**](https://www.codingbroz.com/java-priority-queue-solution/)

## ****Solution – Java Reflection – Attributes****

* [**Can You Access? – Hacker Rank Solution**](https://www.codingbroz.com/can-you-access-solution/)
* [**Java Singleton Pattern – Hacker Rank Solution**](https://www.codingbroz.com/java-singleton-pattern-solution/)
* [**Java Visitor Pattern – Hacker Rank Solution**](https://www.codingbroz.com/java-visitor-pattern-solution/)
* [**Java Annotations – Hacker Rank Solution**](https://www.codingbroz.com/java-annotations-solution/)