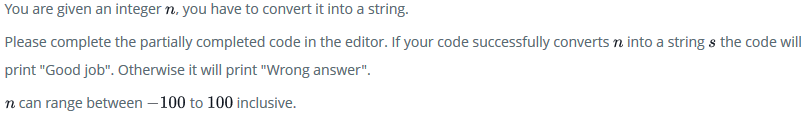
**Java Int to String**(Easy)



Case link: <https://www.hackerrank.com/challenges/java-int-to-string/problem>

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

        String 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();

    }

   }

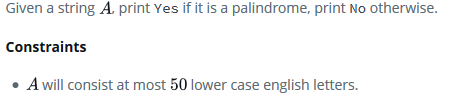
  };

  System.setSecurityManager(securityManager);

 }

}

# Java String Reverse(Easy)



Caselink:

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. \*/

        if(A.equals(new StringBuilder(A).reverse().toString())){

    System.out.println("Yes");

}else{

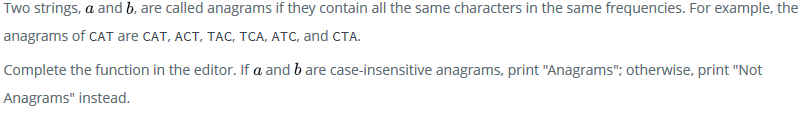
    System.out.println("No");

}

    }

}

# Java Anagrams(Easy)



Case link: <https://www.hackerrank.com/challenges/java-anagrams/problem>

import java.util.Scanner;

public class Solution {

    static boolean isAnagram(String a, String b) {

        // Complete the function

        if(a.length() != b.length()){

            return false;

        }

        for(String s : a.split("")){

            b = b.replaceFirst(s.toLowerCase()+"|"+s.toUpperCase(),"");

        }

        return b.isEmpty();

    }

    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" );

    }

}

# Java String Tokens(Easy)

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

**Note:** You may find the [String.split](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-) method helpful in completing this challenge.

Case link: <https://www.hackerrank.com/challenges/java-string-tokens/problem>

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();

        // Write your code here.

        scan.close();

        s = s.trim();

        if(s.length()== 0){

            System.out.println(0);

            return;

        }

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

        for(int j=1; j<=1; j++){

            System.out.println(ar.length);

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

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

        }

    }

    }

}

# Pattern Syntax Checker(Easy)

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

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

**Note**: In this problem, a regex is only valid if you can compile it using the [Pattern.compile](http://docs.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#compile%28java.lang.String%29) method.

Case link: <https://www.hackerrank.com/challenges/pattern-syntax-checker/problem>

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();

            //Write your code

              try{

                        Pattern.compile(pattern);

                        System.out.println("Valid");

                    }

                    catch(PatternSyntaxException ex){

                        System.out.println("Invalid");

                    }

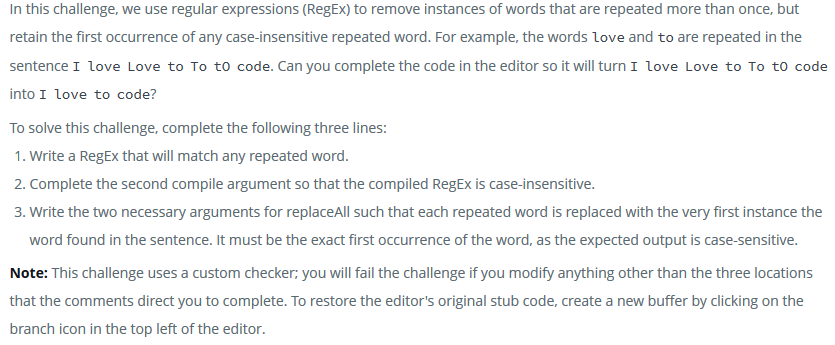
                    testCases--;

        }

    }

}

# Java Regex 2 - Duplicate Words(Medium)



Case link: <https://www.hackerrank.com/challenges/duplicate-word/problem>

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);//biar 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);

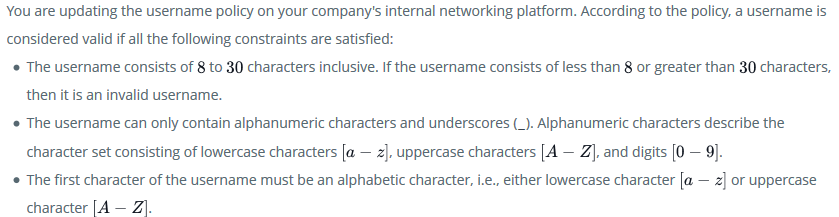
        }

        in.close();

    }

}

# Valid Username Regular Expression(Easy)



Case link: <https://www.hackerrank.com/challenges/valid-username-checker/problem>

import java.util.Scanner;

import java.util.regex.\*;

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");

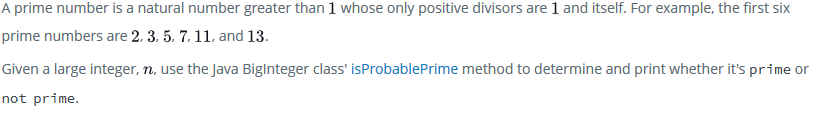
            }

        }

    }

}

# Java Primality Test(Easy)



Case link: <https://www.hackerrank.com/challenges/java-primality-test/problem>

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) {

        String n = scanner.nextLine();

        BigInteger primary = new BigInteger(n);

        if(primary.isProbablePrime(1)){

            System.out.println("prime");

        }else{

            System.out.println("not prime");

        }

        scanner.close();

    }

}

# Java BigInteger(Easy)

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

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

Case link: <https://www.hackerrank.com/challenges/java-biginteger/problem>

import java.io.\*;

import java.util.\*;

import java.math.BigInteger;

import java.util.Scanner;

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 input = new Scanner(System.in);

        BigInteger a = new BigInteger(input.next());

        BigInteger b = new BigInteger(input.next());

        System.out.println(a.add(b));

        System.out.println(a.multiply(b));

    }

}