**Mobile Application Development**

* **Implement Hacker Rank Java Code:**

# Welcome to Java!

# CODE:

public class Solution {

    public static void main(String[] args) {

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

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

    }

}

# Java Stdin and Stdout I

# CODE:

import java.util.Scanner;

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();

        scan.close();

        System.out.println(a);

        System.out.println(b);

        System.out.println(c);

    }

}

# Java Stdin and Stdout II

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int i    = scan.nextInt();

        double d = scan.nextDouble();

        scan.nextLine();              // gets rid of the pesky newline

        String s = scan.nextLine();

        scan.close();

        System.out.println("String: " + s);

        System.out.println("Double: " + d);

        System.out.println("Int: " + i);

    }

}

# Java If-Else

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        scan.close();

        String ans = "";

        if (n % 2 == 1) {

            ans = "Weird";

        } else {

            if (n >= 6 && n <= 20) {

                ans = "Weird";

            } else {

                ans = "Not Weird";

            }

        }

        System.out.println(ans);

    }

}

# Java Output Formatting

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

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

        for (int i = 0; i < 3; i++) {

            String s1 = scan.next();

            int x = scan.nextInt();

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

        }

        scan.close();

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

    }

}

# Java Loops I

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int multiplier = scan.nextInt();

        scan.close();

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

            System.out.format("%d x %d = %d%n", multiplier, i, i \* multiplier);

        }

    }

}

# Java Loops II

# CODE:

import java.util.Scanner;

class Solution{

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int t = scan.nextInt();

        for (int i = 0; i < t; i++) {

            int a = scan.nextInt();

            int b = scan.nextInt();

            int n = scan.nextInt();

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

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

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

            }

            System.out.println();

        }

        scan.close();

    }

}

# Java Datatypes

# CODE:

import java.util.Scanner;

class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int t = scan.nextInt();

        for (int i = 0; i < t; i++) {

            try {

                long x = scan.nextLong();

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

                if (x >= Byte.MIN\_VALUE && x <= Byte.MAX\_VALUE) {

                    System.out.println("\* byte");

                }

                if (x >= Short.MIN\_VALUE && x <= Short.MAX\_VALUE) {

                    System.out.println("\* short");

                }

                if (x >= Integer.MIN\_VALUE && x <= Integer.MAX\_VALUE) {

                    System.out.println("\* int");

                }

                if (x >= Long.MIN\_VALUE && x <= Long.MAX\_VALUE) {

                    System.out.println("\* long");

                }

            } catch (Exception e) {

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

            }

        }

        scan.close();

    }

}

# Java End-of-file

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int i = 1;

        while (scan.hasNextLine()) {

            System.out.println(i + " " + scan.nextLine());

            i++;

        }

        scan.close();

    }

}

# Java Static Initializer Block

# CODE:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class Solution {

    private static int B;

    private static int H;

    private static boolean flag;

    static {

        Scanner scan = new Scanner(System.in);

        B = scan.nextInt();

        H = scan.nextInt();

        scan.close();

        if (B <= 0 || H <= 0) {

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

            flag = false;

        } else {

            flag = true;

        }

    }

    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

# CODE:

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 = String.valueOf(n);

   //Write your code here

   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

# CODE:

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;

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) {

        int d = Integer.valueOf(day);

        int m = Integer.valueOf(month);

        int y = Integer.valueOf(year);

        LocalDate date = LocalDate.of(y, m, d);

        return date.getDayOfWeek().toString();

    }

}

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

# CODE:

import java.util.Scanner;

import java.text.NumberFormat;

import java.util.Locale;

public class Solution {

    public static void main(String[] args) {

        /\* Save input \*/

        Scanner scan = new Scanner(System.in);

        double payment = scan.nextDouble();

        scan.close();

        Locale indiaLocale = new Locale("en", "IN");

        NumberFormat us     = NumberFormat.getCurrencyInstance(Locale.US);

        NumberFormat india  = NumberFormat.getCurrencyInstance(indiaLocale);

        NumberFormat china  = NumberFormat.getCurrencyInstance(Locale.CHINA);

        NumberFormat france = NumberFormat.getCurrencyInstance(Locale.FRANCE);

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

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

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

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

    }

}

# String Introduction

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String A = scan.next();

        String B = scan.next();

        scan.close();

        System.out.println(A.length() + B.length());

       System.out.println(A.compareTo(B) > 0 ? "Yes": "No");

        System.out.println(capFirstLetter(A) + " " + capFirstLetter(B));

    }

    private static String capFirstLetter(String str) {

        if (str == null || str.length() == 0) {

            return "";

        } else {

            return str.substring(0,1).toUpperCase() + str.substring(1);

        }

    }

}

# Substring

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        String str = scan.next();

        int start  = scan.nextInt();

        int end    = scan.nextInt();

        scan.close();

        System.out.println(str.substring(start, end));

    }

}

# Substring comparisons

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        /\* Save input \*/

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        int k    = scan.nextInt();

        scan.close();

        /\* Create smallest and largest strings and initialize them \*/

        String smallest = s.substring(0, k);

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

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

            String curr = s.substring(i, i + k);

            if (smallest.compareTo(curr) > 0){

                smallest = curr;

            }

            if (largest.compareTo(curr) < 0) {

                largest = curr;

            }

        }

        /\* Print results \*/

        System.out.println(smallest);

        System.out.println(largest);

    }

}

# String reverse

# CODE:

import java.util.Scanner;

/\* If a String is equivalent to itself when reversed, it's a palindrome \*/

public class Solution {

    public static void main(String[] args) {

        /\* Read input \*/

        Scanner scan = new Scanner(System.in);

        String str = scan.nextLine();

        scan.close();

        /\* Reverse string and compare to original \*/

        String reversed = new StringBuilder(str).reverse().toString();

        System.out.println(str.equals(reversed) ? "Yes" : "No");

    }

}

# Anagrams

# CODE:

import java.io.\*;

import java.util.\*;

public class Solution {

    //  Time Complexity: O(n) using a HashMap

    // Space Complexity: O(n)

    static boolean isAnagram(String a, String b) {

        if (a == null || b == null || a.length() != b.length()) {

            return false;

        }

        a = a.toLowerCase();

        b = b.toLowerCase();

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

        /\* Fill HashMap with 1st String \*/

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

            char ch = a.charAt(i);

            map.merge(ch, 1, Integer::sum);

        }

        /\* Compare 2nd String to 1st String's HashMap \*/

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

            char ch = b.charAt(i);

            if (map.containsKey(ch) && map.get(ch) > 0) {

                map.put(ch, map.get(ch) - 1);

            } else {

                return false;

            }

        }

        return true;

    }

  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 token

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        /\* Read input \*/

        Scanner scan = new Scanner(System.in);

        String s = scan.nextLine();

        scan.close();

        s = removeLeadingNonLetters(s);

        /\* Check special cases \*/

        if (s.length() == 0) {

            System.out.println(0);

            return;

        }

        /\* Split on all non-alphabetic characters \*/

        String[] words = s.split("[^a-zA-Z]+");

        /\* Print output \*/

        System.out.println(words.length);

        for (String word : words) {

            System.out.println(word);

        }

    }

    private static String removeLeadingNonLetters(String str) {

        int i;

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

            if (Character.isLetter(str.charAt(i))) {

                break;

            }

        }

        return str.substring(i);

    }

}

# Syntax checker

# CODE:

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.PatternSyntaxException;

// If a PatternSyntaxException is not thrown by Pattern.compile, the regular expresion is valid

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int testCases = scan.nextInt();

        scan.nextLine(); // gets rid of the pesky newline.

        while (testCases-- > 0) {

           String pattern = scan.nextLine();

           try {

               Pattern.compile(pattern);

               System.out.println("Valid");

           } catch (PatternSyntaxException exception) {

               System.out.println("Invalid");

           }

        }

        scan.close();

    }

}

# Regex

# CODE:

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.util.Scanner;

class Solution {

    public static void main(String []args) {

        Scanner in = new Scanner(System.in);

        while(in.hasNext()) {

            String IP = in.next();

            System.out.println(IP.matches(new MyRegex().pattern));

        }

    }

}

class MyRegex {

    String num = "([01]?\\d{1,2}|2[0-4]\\d|25[0-5])";

    String pattern = num + "." +  num + "." +  num + "." + num;

}

# Regex-2 duplicate words

# CODE:

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();

    }

}

# Username Regular Expression

# CODE:

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 in = new Scanner(System.in);

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

      while (testCases > 0) {

         String username = in.nextLine();

         String pattern = "^[a-zA-Z]\\w{7,29}$";

         Pattern r = Pattern.compile(pattern);

         Matcher m = r.matcher(username);

         if (m.find()) {

            System.out.println("Valid");

         } else {

            System.out.println("Invalid");

         }

         testCases--;

      }

   }

}

# Tag content extractor

# CODE:

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\* Solution assumes we can't have the symbol "<" as text between tags \*/

public class Solution {

    public static void main(String[] args) {

        Scanner scan  = new Scanner(System.in);

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

        while (testCases-- > 0) {

            String line = scan.nextLine();

            boolean matchFound = false;

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

            Matcher m = r.matcher(line);

            while (m.find()) {

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

                matchFound = true;

            }

            if (!matchFound) {

                System.out.println("None");

            }

        }

        scan.close();

    }

}

# 1D array

# CODE:

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]);

        }

    }

}

# 1D array Part-2

# CODE:

import java.util.\*;

public class Solution {

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

    if (game == null) {

        return false;

    }

    return isSolvable(leap, game, 0);

}

private static boolean isSolvable(int leap, int[] game, int i) {

    // Base Cases

    if (i >= game.length) {

        return true;

    } else if (i < 0 || game[i] == 1) {

        return false;

    }

    game[i] = 1; // marks as visited

    // Recursive Cases (Tries +m first to try to finish game quickly)

    return isSolvable(leap, game, i + leap)

        || isSolvable(leap, game, i + 1)

        || isSolvable(leap, game, i - 1);

}

    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();

    }

}

# 2D array

# CODE:

import java.util.Scanner;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int arr[][] = new int[6][6];

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

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

                arr[row][col] = scan.nextInt();

            }

        }

        scan.close();

        System.out.println(maxHourglass(arr));

    }

    public static int maxHourglass(int [][] arr) {

        int max = Integer.MIN\_VALUE;

        for (int row = 0; row < 4; row++) {

            for (int col = 0; col < 4; col++) {

                int sum = findSum(arr, row, col);

                max = Math.max(max, sum);

            }

        }

        return max;

    }

    private static int findSum(int [][] arr, int r, int c) {

        int sum = arr[r+0][c+0] + arr[r+0][c+1] + arr[r+0][c+2]

                                + arr[r+1][c+1] +

                  arr[r+2][c+0] + arr[r+2][c+1] + arr[r+2][c+2];

        return sum;

    }

}

# Subarray

# CODE:

import java.util.Scanner;

// A subarray must be contiguous. There are O(n^2) contiguous subarrays.

//  Time Complexity: O(n^2)

// Space Complexity: O(1)

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int size     = scan.nextInt();

        int[] array = new int[size];

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

            array[i] = scan.nextInt();

        }

        scan.close();

        System.out.println(negativeSubarrays(array));

    }

    private static int negativeSubarrays(int[] array) {

        int count = 0;

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

            int sum = 0;

            for (int j = i; j < array.length; j++) {

                sum += array[j];

                if (sum < 0) {

                    count++;

                }

            }

        }

        return count;

    }

}

# Arraylist

# CODE:

import java.util.Scanner;

import java.util.ArrayList;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        /\* Save numbers in 2-D ArrayList \*/

        ArrayList<ArrayList<Integer>> lists = new ArrayList();

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

            int d = scan.nextInt();

            ArrayList<Integer> list = new ArrayList();

            for (int col = 0; col < d; col++) {

                list.add(scan.nextInt());

            }

            lists.add(list);

        }

        /\* Answer the queries \*/

        int q = scan.nextInt();

        for (int i = 0; i < q; i++) {

            int x = scan.nextInt();

            int y = scan.nextInt();

            ArrayList<Integer> list = lists.get(x-1);

            if (y <= list.size()) {

                System.out.println(list.get(y-1));

            } else {

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

            }

        }

        scan.close();

    }

}

# List

# CODE:

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);

        }

        /\* Perform 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 + " ");

        }

    }

}

# Map

# CODE:

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

import java.util.HashMap;

class Solution {

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

        /\* Save input as entries in a HashMap \*/

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

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

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

        while (n-- > 0) {

            String name = br.readLine();

            int phone   = Integer.parseInt(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();

    }

}

# Stack

# CODE:

import java.util.Scanner;

import java.util.HashMap;

import java.util.ArrayDeque;

class Solution {

    public static void main(String[] args) {

        /\* Create HashMap to match opening brackets with closing brackets \*/

        HashMap<Character, Character> map = new HashMap();

        map.put('(', ')');

        map.put('[', ']');

        map.put('{', '}');

        /\* Test each expression for validity \*/

        Scanner scan = new Scanner(System.in);

        while (scan.hasNext()) {

            String expression = scan.next();

            System.out.println(isBalanced(expression, map) ? "true" : "false" );

        }

        scan.close();

    }

    private static boolean isBalanced(String expression, HashMap<Character, Character> map) {

        if ((expression.length() % 2) != 0) {

            return false; // odd length Strings are not balanced

        }

        ArrayDeque<Character> deque = new ArrayDeque(); // use deque as a stack

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

            Character ch = expression.charAt(i);

            if (map.containsKey(ch)) {

                deque.push(ch);

            } else if (deque.isEmpty() || ch != map.get(deque.pop())) {

                return false;

            }

        }

        return deque.isEmpty();

    }

}

# Hashset

# CODE:

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();

        }

        s.close();

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

        for (int i = 0; i < t; i++) {

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

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

        }

    }

}

# Generics

# CODE:

    import java.io.IOException;

    import java.lang.reflect.Method;

    class Printer {

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

            for (T item : array) {

                System.out.println(item);

            }

        }

    }

    public class Solution {

        public static void main(String args[]) {

            Printer myPrinter    = new Printer();

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

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

            myPrinter.printArray(intArray);

            myPrinter.printArray(stringArray);

            int count = 0;

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

                String name = method.getName();

                if (name.equals("printArray")) {

                    count++;

                }

            }

            if (count > 1) {

                System.out.println("Method overloading is not allowed!");

            }

        }

    }

# Comparator

# CODE:

import java.util.\*;

class Checker implements Comparator<Player> {

    @Override

    public int compare(Player p1, Player p2) {

        if (p1.score == p2.score) {

            return p1.name.compareTo(p2.name);

        } else {

            return p2.score - p1.score; // descending order

        }

    }

}

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);

        }

    }

}

# Sort

# CODE:

import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

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;

    }

}

public class Solution {

    public static void main(String[] args) {

        Scanner scan  = new Scanner(System.in);

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

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

        while (testCases-- > 0) {

            int id       = scan.nextInt();

            String fname = scan.next();

            double cgpa  = scan.nextDouble();

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

            studentList.add(st);

        }

        scan.close();

        Collections.sort(studentList, new StudentComparator());

        for (Student st: studentList) {

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

        }

    }

}

class StudentComparator implements Comparator<Student> {

    double epsilon = 0.001; // since we shouldn't use "==" with doubles

    @Override

    public int compare(Student s1, Student s2) {

        if (Math.abs(s1.getCgpa() - s2.getCgpa()) > epsilon) {

            return s1.getCgpa() < s2.getCgpa() ? 1 : -1; // descending order

        } else if (!s1.getFname().equals(s2.getFname())) {

            return s1.getFname().compareTo(s2.getFname());

        } else {

            return s1.getId() - s2.getId();

        }

    }

}

# Dequeue

# CODE:

import java.util.Scanner;

import java.util.ArrayDeque;

import java.util.HashMap;

public class test {

    public static void main(String[] args) {

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

        ArrayDeque<Integer> deque     = new ArrayDeque();

        Scanner scan = new Scanner(System.in);

        int n = scan.nextInt();

        int m = scan.nextInt();

        int max = 0;

        for (int i = 0; i < n; i++) {

            /\* Remove old value (if necessary) \*/

            if (i >= m) {

                int old = deque.removeFirst();

                if (map.get(old) == 1) {

                    map.remove(old);

                } else {

                    map.merge(old, -1, Integer::sum);

                }

            }

            /\* Add new value \*/

            int num = scan.nextInt();

            deque.addLast(num);

            map.merge(num, 1, Integer::sum);

            max = Math.max(max, map.size());

            if (max == m) {

                break;

            }

        }

        scan.close();

        System.out.println(max);

    }

}

# Bitset

# CODE:

import java.util.Scanner;

import java.util.BitSet;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        int N = scan.nextInt();

        int M = scan.nextInt();

        BitSet B1 = new BitSet(N);

        BitSet B2 = new BitSet(N);

        while (M-- > 0) {

            String str = scan.next();

            int a      = scan.nextInt();

            int b      = scan.nextInt();

            switch (str) {

                case "AND":

                    if (a == 1) {

                        B1.and(B2);

                    } else {

                        B2.and(B1);

                    }

                    break;

                case "OR":

                    if (a == 1) {

                        B1.or(B2);

                    } else {

                        B2.or(B1);

                    }

                    break;

                case "XOR":

                    if (a == 1) {

                        B1.xor(B2);

                    } else {

                        B2.xor(B1);

                    }

                    break;

                case "FLIP":

                    if (a == 1) {

                        B1.flip(b);

                    } else {

                        B2.flip(b);

                    }

                    break;

                case "SET":

                    if (a == 1) {

                        B1.set(b);

                    } else {

                        B2.set(b);

                    }

                    break;

                default:

                    break;

            }

            System.out.println(B1.cardinality() + " " + B2.cardinality());

        }

        scan.close();

    }

}

# Priority Queue

# CODE:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

import java.util.Comparator;

import java.util.PriorityQueue;

class Student {

    private final int id;

    private final String name;

    private final double cgpa;

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

        this.id = id;

        this.name = name;

        this.cgpa = cgpa;

    }

    public int getID() {

        return id;

    }

    public String getName() {

        return name;

    }

    public double getCGPA() {

        return cgpa;

    }

}

class Priorities {

    private final PriorityQueue<Student> queue = new PriorityQueue<>(

            Comparator.comparing(Student::getCGPA).reversed()

                    .thenComparing(Student::getName)

                    .thenComparing(Student::getID));

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

        events.forEach((event) -> {

            if (event.equals("SERVED")) {

                queue.poll();

            } else {

                String[] details = event.split(" ");

                queue.add(new Student(Integer.parseInt(details[3]), details[1], Double.parseDouble(details[2])));

            }

        });

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

        while (!queue.isEmpty()) {

            students.add(queue.poll());

        }

        return students;

    }

}

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());

            }

        }

    }

}

# Inheritance I

# CODE:

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();

   }

}

# Inheritance II

# CODE:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

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");

    }

}

# Abstract Class

# CODE:

import java.util.\*;

abstract class Book {

    String title;

    abstract void setTitle(String s);

    String getTitle() {

        return title;

    }

}

class MyBook extends Book {

    @Override

    void setTitle(String s) {

        title = s;

    }

}

public class Main {

    public static void main(String []args) {

        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();

    }

}

# Interface

# CODE:

import java.util.\*;

interface AdvancedArithmetic {

    int divisor\_sum(int n);

}

class MyCalculator implements AdvancedArithmetic {

    public int divisor\_sum(int n) {

        int sum  = 0;

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

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

            if (n % i == 0) { // if "i" is a divisor

                sum += i + n/i; // add both divisors

            }

        }

        /\* If sqrt is a divisor, we should only count it once \*/

        if (sqrt \* sqrt == n) {

            sum -= sqrt;

        }

        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();

    }

    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);

        }

    }

}

# Method Overriding

# CODE:

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";

    }

    @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();

    }

}

# Method Overriding 2

# CODE:

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 = super.define\_me();

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

    }

}

class Solution {

    public static void main(String[] args) {

        MotorCycle M = new MotorCycle();

    }

}

# Instance of keyword

# CODE:

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));

    }

}

# Iterator

# CODE:

import java.util.\*;

public class Main{

   static Iterator func(ArrayList mylist){

      Iterator it=mylist.iterator();

      while(it.hasNext()){

         Object element = it.next();

         if (element.equals("###"))

            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);

      }

   }

}

# Exception Handling

# CODE:

import java.util.\*;

import java.util.Scanner;

class MyCalculator {

    long power(int n, int p) throws Exception {

        if (n < 0 || p < 0) {

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

        } else if (n == 0 && p == 0) {

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

        } else {

            return (long) Math.pow(n, p);

        }

    }

}

class Solution {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        while (in .hasNextInt()) {

            int n = in .nextInt();

            int p = in .nextInt();

            MyCalculator my\_calculator = new MyCalculator();

            try {

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

            } catch (Exception e) {

                System.out.println(e);

            }

        }

    }

}

# Exception Handling (Try Catch)

# CODE:

import java.util.Scanner;

import java.util.InputMismatchException;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

        try {

            int x = scan.nextInt();

            int y = scan.nextInt();

            System.out.println(x/y);

        } catch (InputMismatchException e) {

            System.out.println(e.getClass().getName());

        } catch (ArithmeticException e) {

            System.out.println(e.getClass().getName() + ": / by zero");

        }

    }

}