



The 2023 ICPC Caribbean Finals Qualifier

Real Contest Problem Set

Problem set developers

Anier Velasco Sotomayor – Harbour Space University
Humberto Yusta Gómez – Harbour Space University
Carlos Joa Fong – Altice Dominicana
Marcelo J. Fornet Fornés – ICPC Caribbean
Alben Luis Urquiza Rojas – Universidad de Las Tunas
Ernesto Teruel Velazco – Universidad de las Ciencias Informáticas
Rubén Alcolea Núñez – Leil Storage

Matcom Online Grader

Leandro Castillo Valdés – ICPC Caribbean

September 23th, 2023

Problem A. Array Distribution

Let A be an array of N integers between 1 and 10. Distribute the numbers in A into two groups G_0 and G_1 satisfying the following conditions:

1. Each group is non empty.
2. Each number in A belongs to either G_0 or G_1 .
3. The product of the numbers in G_0 equals the product of the numbers in group G_1 .

Input

The first line of the input is an integer N ($2 \leq N \leq 3000$). The second line contains N integers A_i ($1 \leq A_i \leq 10$) for $1 \leq i \leq N$.

Output

If it is not possible to distribute the numbers in array A satisfying the above conditions, print the word “**IMPOSSIBLE**” in a single line. Otherwise, print the word “**POSSIBLE**” on the first line, followed by a line consisting of a binary string S of N digits, where the i -th digit $S_i = 0$ if A_i belongs to group G_0 or $S_i = 1$ if it belongs to group G_1 . If there are multiple ways to distribute array A , you may print any binary string corresponding to that valid distribution.

Example

standard input	standard output
5 1 2 3 4 6	POSSIBLE 00110
3 3 2 1	IMPOSSIBLE

Problem B. Broken Watch

Bill was very happy because his mom bought him a beautiful watch for his birthday. However, his happiness was short-lived because while he was playing with his cat, it pushed the watch trying to escape and the watch fell to the floor.

After this incident, his watch doesn't work well: the seconds hand walks $k/60$ units per second. Notice that $k = 1$ means that the watch is working correctly!

Bill is a little worried about this situation and asked your help to solve the following problem: if he were to start his watch and a new one working correctly at 12:00am, how long does it take for both watches to match their seconds hands exactly n times? As Bill dislikes non-integer numbers, he asks you to ignore matches that occur on non-integer number of seconds (eg 1.5s).

Input

The first line of the input contains an integer T ($1 \leq T \leq 100$), the number of tests cases. The next T lines have two integers n ($1 \leq n \leq 10^9$) and k ($0 \leq k \leq 59$) separated by a whitespace, representing the number of times the seconds hands of both watches need to match exactly and the deviation of Bill's watch in units per second respectively.

Output

The output contains T lines with the answer for each test case: the number of seconds since 12:00am when the n -th match occurs for the given deviation k .

Example

standard input	standard output
2	300
5 2	150
5 3	
3	31560
1578 40	1234560
123456 55	599999940
9999999 30	

Problem C. Cutting Rebar

A building company is designing a new construction work. The foundation requires gravel, cement, water, sand, and rebar. However, the workers have already wasted a large portion of the rebar due to their lack of knowledge. Therefore, the ZETI company needs a program to determine the minimum number of rebar needed for the project and how the rebar should be cut.

Initially, there are infinitely many rebar rods of length M . But small fragments of rebar are needed, so it is necessary to cut the larger pieces to obtain the smaller ones. Thus, the same rebar rod can be cut multiple times to obtain smaller pieces.

Write a program to help ZETI company to know how many large rebar rods are initially needed to obtain all the small pieces?

Input

The first input line has two integers N ($1 \leq N$) and M ($1 \leq M \leq 100$) separated by a whitespace, representing the number of lines that follow and the length of the available rebar at the beginning of the construction.

The following N lines contain an integer C_i ($1 \leq C_i$) and a real number L_i ($0 < L_i \leq M$) separated by a whitespace. Each line means that C_i pieces of length L_i are required for the new construction work. It is always guaranteed that L_i will have exactly two digits after the decimal point and $\sum_1^N C_i \leq 18$ for all test cases.

Output

The output should contain the minimum number of large rebar needed to obtain the smaller pieces.

Example

standard input	standard output
2 32 1 20.99 4 9.21	2
6 14 1 10.33 1 3.24 1 12.51 1 13.94 1 3.12 1 9.15	4

Hints:

In the second test case the solution could be:

(10.33 + 3.24), (12.51), (13.94), (3.12 + 9.15)

For a total of 4 large rebar.

Problem D. Dragonslayers

A long time ago, in the kingdom of Berland, there are three dragons that are terrorizing the population. As the king of Berland, you enlisted the help of three knights and ordered them to slay the dragons.

Each knight is numbered from 1 to 3. Likewise, the dragons are numbered from 1 to 3.

Knight i has a certain attack skill encoded as a positive integer value A_i . On the other hand, dragon j has a certain defensive strength, also encoded as a positive integer value D_j . For knight i to kill dragon j , the knight's attack skill value A_i must be strictly greater than to the dragon's defensive strength D_j .

As the dragons live far apart from each other, you must assign exactly one dragon to each knight. This knight will try his best to kill his assigned dragon.

As you are aware of how mighty these dragons are, you consider your campaign to be successful if the knights are able to kill at least two of the three dragons.

Determine if it is possible to have a successful campaign and an assignment of dragons to knights to achieve a successful campaign.

Input

The input consists of two lines in the following format

A_1 A_2 A_3
 D_1 D_2 D_3

where A_1 , A_2 , A_3 are the attack skill values of knights 1, 2 and 3, respectively. And D_1 , D_2 , D_3 are the defensive strength values of dragons 1, 2 and 3, respectively

All values are integers between 1 and 9.

Output

If it is not possible to have a successful campaign, print a single line with the word “**IMPOSSIBLE**”.

Otherwise, print two lines, where the first line consists of the word “**POSSIBLE**” and the second line has three integers X_1 , X_2 and X_3 , where X_i is the id of the dragon assigned to knight i .

Example

standard input	standard output
9 1 3 2 5 4	POSSIBLE 2 3 1
1 2 3 2 2 2	IMPOSSIBLE

Problem E. Stolen Table

Alice had an $N \times M$ table of positive integers. Bob decided to play a trick on Alice and stole the table. But he told Alice what the maximum value is in each row and each column of the table. Bob will only return the table if Alice can tell him how many different tables can have the given maximum values. Help Alice to recover her table.

Input

In the first input line, two integers n and m ($1 \leq n, m \leq 2 \cdot 10^5$) are given, representing the size of the table.

In the second input line, there are n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$) - the maximum values in each row.

In the third input line, there are m integers b_1, b_2, \dots, b_m ($1 \leq b_j \leq 10^9$) - the maximum values in each column.

Output

Output a single integer - the number of different tables that satisfy Bob's conditions. Since the answer can be very large, print it modulo $10^9 + 7$.

Example

standard input	standard output
3 3 2 2 3 2 3 3	89
1 1 1 2	0
5 5 2 2 3 3 3 2 2 2 3 3	49049891
12 13 2 2 2 3 3 4 4 4 4 5 5 5 2 3 3 3 3 4 5 5 5 5 5 5 5	808346164
2 3 2 3 3 1 5	0

Problem F. Fibonacci Squared

In this problem you will be playing with a slight different version of the very known Fibonacci sequence. We call it Fibonacci Squared because the most relevant difference compared to the original sequence is the addition of squared term determined by the index of n th term to be computed.

The Fibonacci Squared looks like this: $F_i = F_{i-1} + F_{i-2} + 2 \cdot i^2 + 5$. This new sequence has the same base cases as the original Fibonacci sequence: $F_0 = 0$ and $F_1 = 1$. The output can be very big, so print the value F_n modulo 1000000007.

Input

The first line contains an integer q ($1 \leq q \leq 5000$) representing the number of terms to be computed. The next q lines contain an integer n ($1 \leq n \leq 10^{18}$) with the n th term to be computed.

Output

The output contains q lines representing the value of each term F_n modulo 1000000007.

Example

standard input	standard output
4	14
2	38
3	182
5	633
7	
4	3233
10	568233175
100	98340106
1000	623251609
10000	

Hints:

The explanation of the first input sample is shown below:

$$Fib[0] = 0$$

$$Fib[1] = 1$$

$$Fib[2] = 1 + 2 \cdot 4 + 5 = \mathbf{14}$$

$$Fib[3] = 14 + 1 + 2 \cdot 9 + 5 = \mathbf{38}$$

$$Fib[4] = 38 + 14 + 2 \cdot 16 + 5 = 89$$

$$Fib[5] = 89 + 38 + 2 \cdot 25 + 5 = \mathbf{182}$$

$$Fib[6] = 182 + 89 + 2 \cdot 36 + 5 = 348$$

$$Fib[7] = 348 + 182 + 2 \cdot 49 + 5 = \mathbf{633}$$

Problem G. Good Subarrays

Given an array a of n integers, a_1, a_2, \dots, a_n . Count the number of odd length subarrays such that the element in the middle of the subarray is strictly bigger than the rest of the elements of the subarray, and contain more even numbers than odd numbers.

Formally, count the number subarrays a_l, a_{l+1}, \dots, a_r such that:

- $r - l + 1$ is odd
- The middle element a_m , where $m = \frac{l+r}{2}$ is strictly greater than all other elements in the subarray a_l, a_{l+1}, \dots, a_r .
- The number of even integers within the subarray a_l, a_{l+1}, \dots, a_r is strictly greater than the number of odd integers.

Input

The first line contains an integer n ($1 \leq n \leq 3 \cdot 10^5$) — the number of elements.

The second line consists of n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$), where a_i is the value of the i -th element.

Output

Print a single integer in a line — the number of subarrays of odd length such that the middle element is strictly bigger than all the others and contains more even elements than odd ones.

Example

standard input	standard output
5 2 4 6 3 3	5
7 4 8 6 5 5 12 8	7

Problem H. Hidden Sequence

There is a sequence of integers c_1, c_2, \dots, c_k . This sequence was hidden, by constructing a new sequence b_1, b_2, \dots, b_{k-1} according to the following rule: each element b_i is equal to the sum of c_i and c_{i+1} (i.e., $b_i = c_i + c_{i+1}$) for all i from 1 to $k - 1$.

Your task is to determine the maximum number of distinct numbers that can be in the original sequence c_1, c_2, \dots, c_k , given the values of b_1, b_2, \dots, b_{k-1} .

To make the task more challenging for reconstructing the sequence c , you receive an array a_1, a_2, \dots, a_n . You need to process m queries in the form of intervals $[l, r]$, where l and r are some indices of elements in the sequence a . For each query, you are required to determine the number of distinct numbers in the sequence c if the sequence b were equal to a_l, a_{l+1}, \dots, a_r .

Input

In the first line, two integers n and m ($1 \leq n \leq 2 \cdot 10^5, 1 \leq m \leq 2 \cdot 10^5$) are entered - the size of the sequence a and the number of queries, respectively.

In the second line, n integers a_1, a_2, \dots, a_n ($-10^9 \leq a_i \leq 10^9$) are entered - the elements of the sequence a .

Then, m lines follow, each containing two integers l and r ($1 \leq l \leq r \leq n$) - the starting and ending indices of the interval for each query.

Output

For each query, output in a separate line the number of distinct numbers in the sequence c if the sequence b were equal to a_l, a_{l+1}, \dots, a_r .

Example

standard input	standard output
10 7	8
0 0 1 2 10 11 3 101 109 55	7
1 10	4
3 10	7
2 4	5
1 9	5
2 5	5
3 6	
6 10	
15 10	7
10 0 -9 3 3 4 6 -6 -4 -4 -11 -1 1 -2 0	9
3 10	3
3 12	12
5 6	7
3 15	2
10 15	3
6 6	2
3 5	4
1 1	6
12 14	
4 10	

Hints:

- If $b = [1, 2, 3]$, then c might be equal to $[2, -1, 3, 0]$.
- If $b = [0]$, then c might be equal to $[1000, -1000]$.

Problem I. Sorting by ASCII

ASCII is the acronym of American Standard Code for Information Interchange. It is the most common character encoding format for text data in computers and Internet. Characters in ASCII encoding include uppercase and lowercase letters (A..Z, a..z), numerals (0..9) and basic punctuation symbols.

In this problem, we are only interested in the ASCII codes of uppercase and lowercase letters of the English alphabet. ASCII defines a code for each letter. Uppercase letters ('A'..'Z') have assigned codes starting from 65 for letter 'A', 66 for letter 'B' and so on until 90 is assigned to letter 'Z'. On the other hand, lowercase letters ('a'..'z') have assigned codes starting from 97 for letter 'a', 98 for letter 'b' and so on until 122 is assigned to letter 'z'.

Knowing the ASCII codes it's possible to define the ASCII sum for a given string as the sum of ASCII codes of its letters. For instance, the ASCII sum for the string 'ABC' = 198 (65 + 66 + 67), while the ASCII sum for the string 'abD' = 263 (97 + 98 + 68).

Given a list of N strings, sort the list in non-descending order by the ASCII sum of the strings.

Input

The first line contains an integer $1 \leq N \leq 100$, representing the size of the list. The next N lines contain the strings of the list. Each string has at most 100 lowercase and uppercase letters of the English alphabet.

Output

The output contains N lines with the sorted list of strings. If there is a tie with the sum of ASCII codes of two strings, consider the string with the smaller index (in the original list) to be smaller than the string with larger index.

Example

standard input	standard output
4 abD ABC ABCA aaa	ABC abD ABCA aaa
3 cbAdefg Adbcefg gfedbcA	cbAdefg Adbcefg gfedbcA

Hints:

Here are the sums of ASCII codes for the strings in the first sample test case:

ABC: $65 + 66 + 67 = 198$

abD: $97 + 98 + 68 = 263$

ABCA: $65 + 66 + 67 + 65 = 263$

aaa: $97 \cdot 3 = 291$

Problem J. Juan and Odd Numbers

Juan loves numbers and is always up for a mathematical challenge. Today, he has come across an interesting problem. He is given an integer, N , where $1 \leq N \leq 10^9$. The task is to calculate the sum of the last digit of the first N odd numbers.

For example, if N is 4, the first four odd numbers are 1, 3, 5, and 7. The last digits of these numbers are 1, 3, 5, and 7, respectively. The sum of these last digits is $1 + 3 + 5 + 7 = 16$.

Juan needs your help to solve this problem. Can you write a program to find the sum of the last digits of the first N odd numbers?

Input

An integer N ($1 \leq N \leq 10^9$), representing the number of odd numbers.

Output

The sum of the last digits of the first N odd numbers. The last digit refers to the rightmost digit of a number.

Example

standard input	standard output
7	29

Hints:

Example case:

First 7 odd numbers:

1, 3, 5, 7, 9, 11, 13

Last digit sum: $1 + 3 + 5 + 7 + 9 + 1 + 3 = 29$.

Problem K. Good Numbers

The proper divisors of a natural number n are the divisors of n that are different from 1 and n .

For example, the proper divisors of 15 are 3 and 5, and the proper divisors of 204 are 2, 3, 4, 6, 12, 17, 34, 51, 68, and 102.

We say a number x is **good** if the sum of its proper divisors is greater than x .

For example:

- 15 is not good because $15 > 3 + 5 = 8$.
- 204 is good because $204 < 2 + 3 + 4 + 6 + 12 + 17 + 34 + 51 + 68 + 102 = 299$

You're given a range $[l, r]$.

Your task is to find the first and the last **good** numbers in the range $[l, r]$. If there are no good numbers in the range $[l, r]$, print the pair "-1 -1".

Input

The only line in the input contains a pair (l, r) , $(1 \leq l \leq r \leq 10^{10})$.

Output

Output a single line, containing two numbers separated by space —the first and last good numbers in the array, respectively. If there aren't any good numbers in the range $[l, r]$, print the pair "-1 -1".

Example

standard input	standard output
204 301	204 300
91 95	-1 -1

Problem L. Distance Graph

In this problem there are n vertices available and k constraints. The i -th constraint specifies that there must be at least a_i nodes at a distance of exactly i from vertex 1.

The goal of this problem is to find the maximum number of undirected and unweighted edges a graph with n vertices can have while satisfying the k constraints. That is, there should be at least a_1 nodes at a distance of 1 from node 1, at least a_2 nodes at a distance of 2 from node 1, and so on. It means at least a_i nodes at a distance of i from node 1, for $1 \leq i \leq k$.

There must not be any edge between two vertices that are at the same distance of vertex 1. For example, if vertices 2 and 3 are both at distance 1 from vertex 1, there must not be any edge between vertices 2 and 3.

The resulting graph must be connected.

If there is no graph with n vertices that meets the constraints, the answer will be -1 .

The distance from vertex a to vertex b is defined as the minimum number of edges required to travel from a to b .

Input

The first line contains two integers n ($3 \leq n \leq 10^9$) and k ($2 \leq k \leq 3 \cdot 10^5$) — the number of vertices the graph should have, and the number of distance constraints.

The second line consists of k integers a_1, a_2, \dots, a_k ($0 \leq a_i \leq 10^9$), where a_i indicates that there must be at least a_i vertices at a distance of i from vertex 1.

It is guaranteed that $a_k \geq 1$.

Output

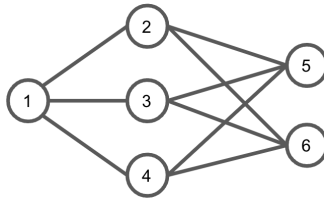
Print a single number on one line — the maximum number of undirected and unweighted edges a graph satisfying the constraints explained above can have, or -1 if there is no such graph.

Example

standard input	standard output
6 2 1 2	9
10 4 1 2 3 4	-1
11 4 1 2 3 4	21
15 4 1 2 3 4	46

Hints:

One possible graph that is a solution for the first example is:



There are three vertices (2, 3 and 4) at distance 1 from vertex 1, so the constraint of having at least 1 vertex at a distance of exactly 1 from vertex 1 is satisfied.

There are two vertices (5 and 6) at distance 2 from vertex 1, so the constraint of having at least 2 vertices at a distance of exactly 2 from vertex 1 is satisfied.

There is no graph of 6 vertices and more than 9 edges that satisfies all the constraints.

Problem M. Minimize the Greatest Value

Bill is in charge of his brother John during the whole day because their mother went out for shopping. Bill studies Computer Sciences and loves very much number theory and algorithms. On the other hand, John is a child and likes very much playing with his brother, specially when it's not the best time :). Today, Bill is participating at the ICPC Caribbean Finals and needs to be focused in the contest.

Bill needs his brother stay busy while he is trying his best performance at the contest. Therefore, Bill assigned the following problem to John. Given a string with N ($1 \leq N \leq 10$) digits from 1 to 9, you need to form K numbers with those digits (which can be reordered) so that the largest of all numbers is minimized.

Input

The first line contains two integers N ($1 \leq N \leq 10$) and K ($1 \leq K \leq N$) separated by a whitespace, representing the number of available digits and the amount of numbers John needs to create respectively. The next line contains a string with the N digits that will be used to create the K numbers.

Output

The output contains a single line with the largest number created.

Example

standard input	standard output
10 10 1234567891	9
6 3 123456	34

Hints:

In the first test case there is only one way to create K numbers, each one with a single digit. So the answer is 9, the largest digit.

In the second test case one solution is (16), (25) and (34), so the largest number is 34. Other valid solution is (15), (26) and (34).