Problem A. Frog 1

Time limit 2000 ms

Problem Statement

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to Stone $i + 1$ or Stone $i + 2$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Constraints

- All values in input are integers.
- $2 \leq N \leq 10^5$
- $1 \leq h_i \leq 10^4$

Input

Input is given from Standard Input in the following format:

```
N
h_1 h_2 ... h_N
```

Output

Print the minimum possible total cost incurred.

Sample 1

Input	Output
4 10 30 40 20	30

If we follow the path $1 \rightarrow 2 \rightarrow 4$, the total cost incurred would be $|10 - 30| + |30 - 20| = 30$.

Sample 2

Input	Output
2 10 10	0

If we follow the path $1 \rightarrow 2$, the total cost incurred would be $|10 - 10| = 0$.

Sample 3

Input	Output
6 30 10 60 10 60 50	40

If we follow the path $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$, the total cost incurred would be $|30 - 60| + |60 - 60| + |60 - 50| = 40$.

Problem B. Frog 2

Time limit 2000 ms

Problem Statement

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to one of the following: Stone $i+1$, $i+2$, $\ldots$, $i+K$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Constraints

- All values in input are integers.
- $2 \leq N \leq 10^5$
- $1 \leq K \leq 100$
- $1 \leq h_i \leq 10^4$

Input

Input is given from Standard Input in the following format:

```
N K
h_1 h_2 ... h_N
```

Output

Print the minimum possible total cost incurred.

Sample 1

Input	Output
5 3 10 30 40 50 20	30

If we follow the path $1 \rightarrow 2 \rightarrow 5$, the total cost incurred would be $|10 - 30| + |30 - 20| = 30$.

Sample 2

Input	Output
3 1 10 20 10	20

If we follow the path $1 \rightarrow 2 \rightarrow 3$, the total cost incurred would be $|10 - 20| + |20 - 10| = 20$.

Sample 3

Input	Output
2 100 10 10	0

If we follow the path $1 \rightarrow 2$, the total cost incurred would be $|10 - 10| = 0$.

Sample 4

Input	Output
10 4 40 10 20 70 80 10 20 70 80 60	40

If we follow the path $1 \rightarrow 4 \rightarrow 8 \rightarrow 10$, the total cost incurred would be $|40 - 70| + |70 - 70| + |70 - 60| = 40$.

Problem C. Vacation

Time limit 2000 ms

Problem Statement

Taro's summer vacation starts tomorrow, and he has decided to make plans for it now.

The vacation consists of N days. For each i ($1 \leq i \leq N$), Taro will choose one of the following activities and do it on the i-th day:

- A: Swim in the sea. Gain a_i points of happiness.
- B: Catch bugs in the mountains. Gain b_i points of happiness.
- C: Do homework at home. Gain c_i points of happiness.

As Taro gets bored easily, he cannot do the same activities for two or more consecutive days.

Find the maximum possible total points of happiness that Taro gains.

Constraints

- All values in input are integers.
- $1 \leq N \leq 10^5$
- $1 \leq a_i, b_i, c_i \leq 10^4$

Input

Input is given from Standard Input in the following format:

N
a_1 b_1 c_1
a_2 b_2 c_2
:
a_N b_N c_N

Output

Print the maximum possible total points of happiness that Taro gains.

Sample 1

Input	Output
3 10 40 70 20 50 80 30 60 90	210

If Taro does activities in the order C, B, C, he will gain $70 + 50 + 90 = 210$ points of happiness.

Sample 2

Input	Output
1 100 10 1	100

Sample 3

Input	Output
7 6 7 8 8 8 3 2 5 2 7 8 6 4 6 8 2 3 4 7 5 1	46

Taro should do activities in the order C, A, B, A, C, B, A.

Problem D. Knapsack 1

Time limit 2000 ms
Mem limit 1048576 kB

Problem Statement

There are N items, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), Item i has a weight of w_i and a value of v_i.

Taro has decided to choose some of the N items and carry them home in a knapsack. The capacity of the knapsack is W, which means that the sum of the weights of items taken must be at most W.

Find the maximum possible sum of the values of items that Taro takes home.

Constraints

- All values in input are integers.
- $1 \leq N \leq 100$
- $1 \leq W \leq 10^5$
- $1 \leq w_i \leq W$
- $1 \leq v_i \leq 10^9$

Input

Input is given from Standard Input in the following format:

N W
w_1 v_1
w_2 v_2
:
w_N v_N

Output

Print the maximum possible sum of the values of items that Taro takes home.

Sample 1

Input	Output
3 8 3 30 4 50 5 60	90

Items 1 and 3 should be taken. Then, the sum of the weights is $3 + 5 = 8$, and the sum of the values is $30 + 60 = 90$.

Sample 2

Input	Output
5 5 1 1000000000 1 1000000000 1 1000000000 1 1000000000 1 1000000000	5000000000

The answer may not fit into a 32-bit integer type.

Sample 3

Input	Output
6 15 6 5 5 6 6 4 6 6 3 5 7 2	17

Items $2, 4$ and 5 should be taken. Then, the sum of the weights is $5 + 6 + 3 = 14$, and the sum of the values is $6 + 6 + 5 = 17$.

Problem E. 1633

Time limit 1000 ms
Mem limit 524288 kB

Your task is to count the number of ways to construct sum n by throwing a dice one or more times. Each throw produces an outcome between 1 and 6.

For example, if $n = 3$, there are 4 ways:

- $1 + 1 + 1$
- $1 + 2$
- $2 + 1$
- 3

Input

The only input line has an integer n.

Output

Print the number of ways modulo $10^9 + 7$.

Constraints

- $1 \leq n \leq 10^6$

Example

Input	Output
3	4

Problem F. Minimizing Coins

Time limit 1000 ms
Mem limit 524288 kB

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal.

For example, if the coins are $\{1, 5, 7\}$ and the desired sum is 11, an optimal solution is $5 + 5 + 1$ which requires 3 coins.

Input

The first input line has two integers n and x: the number of coins and the desired sum of money.

The second line has n distinct integers $c_1, c_2, \ldots, c_n$: the value of each coin.

Output

Print one integer: the minimum number of coins. If it is not possible to produce the desired sum, print -1.

Constraints

- $1 \leq n \leq 100$
- $1 \leq x \leq 10^6$
- $1 \leq c_i \leq 10^6$

Example

Input	Output
3 11 1 5 7	3

Problem G. Coin Combinations I

Time limit 1000 ms
Mem limit 524288 kB

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct ways you can produce a money sum x using the available coins.

For example, if the coins are $\{2, 3, 5\}$ and the desired sum is 9, there are 8 ways:

- $2 + 2 + 5$
- $2 + 5 + 2$
- $5 + 2 + 2$
- $3 + 3 + 3$
- $2 + 2 + 2 + 3$
- $2 + 2 + 3 + 2$
- $2 + 3 + 2 + 2$
- $3 + 2 + 2 + 2$

Input

The first input line has two integers n and x: the number of coins and the desired sum of money.

The second line has n distinct integers $c_1, c_2, \ldots, c_n$: the value of each coin.

Output

Print one integer: the number of ways modulo $10^9 + 7$.

Constraints

- $1 \le n \le 100$
- $1 \le x \le 10^6$
- $1 \le c_i \le 10^6$

Example

Input	Output
3 9 2 3 5	8

Problem H. Coin Combinations II

Time limit 1000 ms
Mem limit 524288 kB

Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct *ordered* ways you can produce a money sum x using the available coins.

For example, if the coins are $\{2, 3, 5\}$ and the desired sum is 9, there are 3 ways:

- $2 + 2 + 5$
- $3 + 3 + 3$
- $2 + 2 + 2 + 3$

Input

The first input line has two integers n and x: the number of coins and the desired sum of money.

The second line has n distinct integers $c_1, c_2, \ldots, c_n$: the value of each coin.

Output

Print one integer: the number of ways modulo $10^9 + 7$.

Constraints

- $1 \le n \le 100$
- $1 \le x \le 10^6$
- $1 \le c_i \le 10^6$

Example

Input	Output
3 9 2 3 5	3

Problem I. Grid Paths

Time limit 1000 ms
Mem limit 524288 kB

Consider an $n \times n$ grid whose squares may have traps. It is not allowed to move to a square with a trap.

Your task is to calculate the number of paths from the upper-left square to the lower-right square. You can only move right or down.

Input

The first input line has an integer n: the size of the grid.

After this, there are n lines that describe the grid. Each line has n characters: `.` denotes an empty cell, and `*` denotes a trap.

Output

Print the number of paths modulo $10^9 + 7$.

Constraints

- $1 \leq n \leq 1000$

Example

Input	Output
4 .... .*.. ...* *...	3

Problem J. Rectangle Cutting

Time limit 1000 ms
Mem limit 524288 kB

Given an $a \times b$ rectangle, your task is to cut it into squares. On each move you can select a rectangle and cut it into two rectangles in such a way that all side lengths remain integers. What is the minimum possible number of moves?

Input

The only input line has two integers a and b.

Output

Print one integer: the minimum number of moves.

Constraints

- $1 \leq a, b \leq 500$

Example

Input	Output
3 5	3

Problem K. Increasing Subsequence

Time limit 1000 ms
Mem limit 524288 kB

You are given an array containing n integers. Your task is to determine the longest increasing subsequence in the array, i.e., the longest subsequence where every element is larger than the previous one.

A subsequence is a sequence that can be derived from the array by deleting some elements without changing the order of the remaining elements.

Input

The first line contains an integer n: the size of the array.

After this there are n integers $x_1, x_2, \ldots, x_n$: the contents of the array.

Output

Print the length of the longest increasing subsequence.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq x_i \leq 10^9$

Example

Input	Output
8 7 3 5 3 6 2 9 8	4

Problem L. Counting Bits

Time limit 1000 ms
Mem limit 524288 kB

Your task is to count the number of one bits in the binary representations of integers between 1 and n.

Input

The only input line has an integer n.

Output

Print the number of one bits in the binary representations of integers between 1 and n.

Constraints

- $1 \leq n \leq 10^{15}$

Example

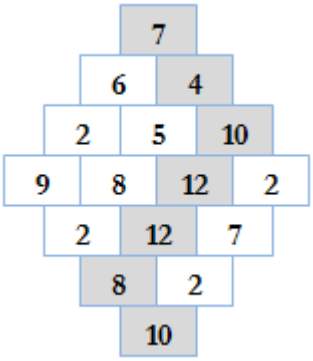
Input	Output
7	12

Explanation: The binary representations of $1 \ldots 7$ are 1, 10, 11, 100, 101, 110, and 111, so there are a total of 12 one bits.

Problem M. Monkey Banana Problem

Time limit 1000 ms
Mem limit 65536 kB

You are in the world of mathematics to solve the great "Monkey Banana Problem". It states that, a monkey enters into a diamond shaped two dimensional array and can jump in any of the adjacent cells **down** from its current position (see figure). While moving from one cell to another, the monkey eats all the bananas kept in that cell. The monkey enters into the array from the upper part and goes out through the lower part. Find the maximum number of bananas the monkey can eat.



Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Every case starts with an integer **N (1 ≤ N ≤ 100)**. It denotes that, there will be **2*N - 1** rows. The **i^{th} (1 ≤ i ≤ N)** line of the next **N** lines contains exactly **i** numbers. Then there will be **N - 1** lines. The **j^{th} (1 ≤ j < N)** line contains **N - j** integers. Each number is greater than zero and less than 2^{15}.

Output

For each case, print the case number and maximum number of bananas eaten by the monkey.

Sample

Input	Output
2 4 7 6 4 2 5 10 9 8 12 2 2 12 7 8 2 10 2 1 2 3 1	Case 1: 63 Case 2: 5

Note

Dataset is huge, use faster I/O methods.

Problem N. Easy Problem

Time limit 2000 ms
Mem limit 262144 kB

Vasya is preparing a contest, and now he has written a statement for an easy problem. The statement is a string of length n consisting of lowercase Latin latters. Vasya thinks that the statement can be considered hard if it contains a subsequence `hard`; otherwise the statement is easy. For example, `hard`, `hzazrzd`, `haaaaard` can be considered hard statements, while `har`, `hart` and `drah` are easy statements.

Vasya doesn't want the statement to be hard. He may remove some characters from the statement in order to make it easy. But, of course, some parts of the statement can be crucial to understanding. Initially the *ambiguity* of the statement is 0, and removing i-th character increases the *ambiguity* by a_i (the index of each character is considered as it was in the original statement, so, for example, if you delete character `r` from `hard`, and then character `d`, the index of `d` is still 4 even though you delete it from the string `had`).

Vasya wants to calculate the minimum *ambiguity* of the statement, if he removes some characters (possibly zero) so that the statement is easy. Help him to do it!

Recall that subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Input

The first line contains one integer n ($1 \le n \le 10^5$) — the length of the statement.

The second line contains one string s of length n, consisting of lowercase Latin letters — the statement written by Vasya.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 998244353$).

Output

Print minimum possible *ambiguity* of the statement after Vasya deletes some (possibly zero) characters so the resulting statement is easy.

Examples

Input	Output
6 hhardh 3 2 9 11 7 1	5

Input	Output
8 hhzarwde 3 2 6 9 4 8 7 1	4

Input	Output
6 hhaarr 1 2 3 4 5 6	0

Note

In the first example, first two characters are removed so the result is `ardh`.

In the second example, 5-th character is removed so the result is `hhzawde`.

In the third example there's no need to remove anything.

Problem O. Yet Another Subarray Problem

Time limit 2000 ms
Mem limit 262144 kB

You are given an array $a_1, a_2, \ldots, a_n$ and two integers m and k.

You can choose some subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$.

The cost of subarray $a_l, a_{l+1}, \ldots, a_{r-1}, a_r$ is equal to $\sum\limits_{i=l}^{r} a_i - k\lceil \frac{r-l+1}{m} \rceil$, where $\lceil x \rceil$ is the least integer greater than or equal to x.

The cost of empty subarray is equal to zero.

For example, if $m = 3$, $k = 10$ and $a = [2, -4, 15, -3, 4, 8, 3]$, then the cost of some subarrays are:

- $a_3 \ldots a_3 : 15 - k\lceil \frac{1}{3} \rceil = 15 - 10 = 5$;
- $a_3 \ldots a_4 : (15 - 3) - k\lceil \frac{2}{3} \rceil = 12 - 10 = 2$;
- $a_3 \ldots a_5 : (15 - 3 + 4) - k\lceil \frac{3}{3} \rceil = 16 - 10 = 6$;
- $a_3 \ldots a_6 : (15 - 3 + 4 + 8) - k\lceil \frac{4}{3} \rceil = 24 - 20 = 4$;
- $a_3 \ldots a_7 : (15 - 3 + 4 + 8 + 3) - k\lceil \frac{5}{3} \rceil = 27 - 20 = 7$.

Your task is to find the maximum cost of some subarray (possibly empty) of array a.

Input

The first line contains three integers n, m, and k ($1 \le n \le 3 \cdot 10^5, 1 \le m \le 10, 1 \le k \le 10^9$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output

Print the maximum cost of some subarray of array a.

Examples

Input	Output
7 3 10 2 -4 15 -3 4 8 3	7

Input	Output
5 2 1000 -13 -4 -9 -20 -11	0

Problem P. New TAP

Time limit	1000 ms
Mem limit	1572864 kB
Code length Limit	50000 B
OS	Linux

We are considering changing the rules for the Argentinian Programming Tournament starting next year. Before doing that, we would like to evaluate whether the new system is fair, and we need your help to do that.

The new tournament will have **N** teams competing in **N-1** rounds. In each round two teams will face each other competing in order to be the first to solve a problem, the losing team being eliminated. In the first round, two teams will be chosen at random, and the losing team will be placed in the last place of the scoreboard, while the winner remains in the competition. In each of the following rounds, two teams still in the competition will be chosen at random, and the losing team shall be placed in the last remaining place of the scoreboard, being thus eliminated from the tournament.

For example, if the tournament has **N=4** teams named "*aWArush*", "*Buen Kilo de Pan Flauta*", "*Melarita*" and "*Type Mismatch*", the tournament will be held in three rounds. Suppose that in the first round "*Buen Kilo de Pan Flauta*" faces "*Melarita*", the former being the winner; in the second round "*aWArush*" beats "*Buen Kilo de Pan Flauta*"; and finally in the last round "*aWArush*" beats "*Type Mismatch*". Then the placement of the teams in the final scoreboard will be in the following order: 1st "*aWArush*", 2nd "*Type Mismatch*", 3rd "*Buen Kilo de Pan Flauta*" and 4th "*Melarita*".

To analize just how fair the new tournament format is, we will consider the teams to be numbered from **1** to **N**, in such a way that lower numbers represent better teams. We will assume that whenever two teams face each other in a given round, the one with the smallest number will invariably win. We would like you to help us answer the following question: What is the probability for team **X** to be placed at position **Y** in the final scoreboard?

Input

There are multiple test cases in the input file. Each test case consists of a single line containing three integer numbers **N**, **X** and **Y**. The number **N** represents the number of teams taking part in the tournament, (**2 ≤ N ≤ 1000**), **X** represents the number of the team we are interested in, and **Y** represents its final position (**1 ≤ X,Y ≤ N**).

Output

For each test case, print a single line containing a rational number, representing the probability for team number **X** to be placed at position **Y** in the final scoreboard. Print the result with exactly **4** digits after the decimal marker, rounding if necessary.

Example

Input	Output
3 2 2 10 3 6 10 1 5 1000 1 1 1000 1000 1000	0.6667 0.0946 0.0000 1.0000 0.0020

Problem Q. Slimes

Time limit 2000 ms
Mem limit 1048576 kB

Problem Statement

There are N slimes lining up in a row. Initially, the i-th slime from the left has a size of a_i.

Taro is trying to combine all the slimes into a larger slime. He will perform the following operation repeatedly until there is only one slime:

- Choose two adjacent slimes, and combine them into a new slime. The new slime has a size of $x + y$, where x and y are the sizes of the slimes before combining them. Here, a cost of $x + y$ is incurred. The positional relationship of the slimes does not change while combining slimes.

Find the minimum possible total cost incurred.

Constraints

- All values in input are integers.
- $2 \leq N \leq 400$
- $1 \leq a_i \leq 10^9$

Input

Input is given from Standard Input in the following format:

```
N
a_1 a_2 ... a_N
```

Output

Print the minimum possible total cost incurred.

Sample 1

Input	Output
4 10 20 30 40	190

Taro should do as follows (slimes being combined are shown in bold):

- (**10**, **20**, 30, 40) → (**30**, 30, 40)
- (**30**, **30**, 40) → (**60**, 40)
- (**60**, **40**) → (**100**)

Sample 2

Input	Output
5 10 10 10 10 10	120

Taro should do, for example, as follows:

- (**10**, **10**, 10, 10, 10) → (**20**, 10, 10, 10)
- (20, **10**, **10**, 10) → (20, **20**, 10)
- (20, **20**, **10**) → (20, **30**)
- (**20**, **30**) → (**50**)

Sample 3

Input	Output
3 1000000000 1000000000 1000000000	5000000000

The answer may not fit into a 32-bit integer type.

Sample 4

Input	Output
6 7 6 8 6 1 1	68

Taro should do, for example, as follows:

- (7, 6, 8, 6, **1**, **1**) → (7, 6, 8, 6, **2**)
- (7, 6, 8, **6**, **2**) → (7, 6, 8, **8**)

- (**7**, **6**, 8, 8) → (**13**, 8, 8)
- (13, **8**, **8**) → (13, **16**)
- (**13**, **16**) → (**29**)

Problem R. Neighbor House

Time limit 1000 ms
Mem limit 65536 kB

The people of Mohammadpur have decided to paint each of their houses red, green, or blue. They've also decided that no two neighboring houses will be painted the same color. The neighbors of house **i** are houses **i-1** and **i+1**. The first and last houses are not neighbors.

You will be given the information of houses. Each house will contain three integers **"R G B"** (quotes for clarity only), where **R, G** and **B** are the costs of painting the corresponding house red, green, and blue, respectively. Return the minimal total cost required to perform the work.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case begins with a blank line and an integer **n (1 ≤ n ≤ 20)** denoting the number of houses. Each of the next **n** lines will contain 3 integers **"R G B"**. These integers will lie in the range **[1, 1000]**.

Output

For each case of input you have to print the case number and the minimal cost.

Sample

Input	Output
2 4 13 23 12 77 36 64 44 89 76 31 78 45 3 26 40 83 49 60 57 13 89 99	Case 1: 137 Case 2: 96

Problem S. Generating Palindromes

Time limit 1000 ms
Mem limit 65536 kB

By definition a string is said to be a palindrome if it does not change when get reversed. `madam` is a nice example of a palindrome.

Given a string **S**, you are allowed to insert any characters at any position of the string, find the minimum number of characters required to make the string a palindrome.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains a string of lowercase letters denoting the string for which we want to generate a palindrome. You may safely assume that the length of the string will be positive and no more than **100**.

Output

For each case, print the case number and the minimum number of characters required to make string to a palindrome.

Sample

Input	Output
6 abcd aaaa abc aab abababaababababa pqrsabcdpqrs	Case 1: 3 Case 2: 0 Case 3: 2 Case 4: 1 Case 5: 0 Case 6: 9

Note

You can easily verify that for a string of length **n**, no more than **(n - 1)** characters are

required to make it a palindrome. For example, **abcd** and its palindrome **abcdcba**.

Problem T. Minimal Coverage

Time limit 1000 ms
Mem limit 262144 kB

You are given n lengths of segments that need to be placed on an infinite axis with coordinates.

The first segment is placed on the axis so that one of its endpoints lies at the point with coordinate 0. Let's call this endpoint the "*start*" of the first segment and let's call its "*end*" as that endpoint that is not the start.

The "start" of each following segment must coincide with the "end" of the previous one. Thus, if the length of the next segment is d and the "end" of the previous one has the coordinate x, the segment can be placed either on the coordinates $[x - d, x]$, and then the coordinate of its "end" is $x - d$, or on the coordinates $[x, x + d]$, in which case its "end" coordinate is $x + d$.

The total *coverage* of the axis by these segments is defined as their overall union which is basically the set of points covered by at least one of the segments. It's easy to show that the coverage will also be a segment on the axis. Determine the minimal possible length of the coverage that can be obtained by placing all the segments on the axis without changing their order.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The next $2t$ lines contain descriptions of the test cases.

The first line of each test case description contains an integer n ($1 \le n \le 10^4$) — the number of segments. The second line of the description contains n space-separated integers a_i ($1 \le a_i \le 1000$) — lengths of the segments in the same order they should be placed on the axis.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

Output

Print t lines, each line containing the answer to the corresponding test case. The answer to a test case should be a single integer — the minimal possible length of the axis coverage.

Examples

Input	Output
6 2 1 3 3 1 2 3 4 6 2 3 9 4 6 8 4 5 7 1 2 4 6 7 7 3 8 8 6 5 1 2 2 3 6	3 3 9 9 7 8

Note

In the third sample test case the segments should be arranged as follows: $[0, 6] \to [4, 6] \to [4, 7] \to [-2, 7]$. As you can see, the last segment $[-2, 7]$ covers all the previous ones, and the total length of coverage is 9.

In the fourth sample test case the segments should be arranged as $[0, 6] \to [-2, 6] \to [-2, 2] \to [2, 7]$. The union of these segments also occupies the area $[-2, 7]$ and has the length of 9.

Problem U. Sugoroku2

Time limit 2000 ms
Mem limit 1048576 kB

Problem Statement

Takahashi is playing sugoroku.

The board has $N + 1$ squares numbered 0 to N. Takahashi starts at Square 0 and head to Square N.

In this sugoroku, we use a wheel showing numbers from 1 through M with equal probability. In each turn, Takahashi spins the wheel and advances by the number shown by the wheel. When it makes him reach Square N or go past it, he wins.

Some of the squares send him to Square 0 when he stops on them. There are K such squares: Square $A_1, \ldots, A_K$.

Find the expected value of the number of times Takahashi spins the wheel before he wins. If it is impossible to win, print `-1` instead.

Constraints

- All values in input are integers.
- $1 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- $0 \leq K \leq 10$
- $0 < A_1 < \ldots < A_K < N$

Input

Input is given from Standard Input in the following format:

N M K
A_1 $\ldots$ A_K

Output

Print the expected value of the number of times Takahashi spins the wheel before he wins. Your output is considered as correct when its absolute or relative error from our answer is at most 10^{-3}. If it is impossible to win, print `-1` instead.

Sample 1

Input	Output
2 2 0	1.5000

If the wheel shows 1 in the first spin, he will need two spins to win; if the wheel shows 2 in the first spin, he will need one spin to win. Thus, the expected number of spins is 1.5.

Sample 2

Input	Output
2 2 1 1	2.0000

If the wheel shows 1, he advances to Square 1, but it sends him back to Square 0.
Thus, he keeps spinning the wheel until he gets 2 and wins.
The probability that he gets 2 for the first time in the i-th spin is $\frac{1}{2^i}$, so the expected number of spins is $\sum_{i=1}^{\infty}(i \times \frac{1}{2^i}) = 2$.

Sample 3

Input	Output
100 6 10 11 12 13 14 15 16 17 18 19 20	-1

Sample 4

Input	Output
100000 2 2 2997 92458	201932.2222

Problem V. Fish

Time limit	3000 ms
Mem limit	131072 kB
Input file	stdin
Output file	stdout

n fish, numbered from 1 to n, live in a lake. Every day right one pair of fish meet, and the probability of each other pair meeting is the same. If two fish with indexes i and j meet, the first will eat up the second with the probability a_{ij}, and the second will eat up the first with the probability $a_{ji} = 1 - a_{ij}$. The described process goes on until there are at least two fish in the lake. For each fish find out the probability that it will survive to be the last in the lake.

Input

The first line contains integer n ($1 \le n \le 18$) — the amount of fish in the lake. Then there follow n lines with n real numbers each — matrix a. a_{ij} ($0 \le a_{ij} \le 1$) — the probability that fish with index i eats up fish with index j. It's guaranteed that the main diagonal contains zeros only, and for other elements the following is true: $a_{ij} = 1 - a_{ji}$. All real numbers are given with not more than 6 characters after the decimal point.

Output

Output n space-separated real numbers accurate to not less than 6 decimal places. Number with index i should be equal to the probability that fish with index i will survive to be the last in the lake.

Examples

Input	Output
2 0 0.5 0.5 0	0.500000 0.500000

Input	Output
5 0 1 1 1 1 0 0 0.5 0.5 0.5 0 0.5 0 0.5 0.5 0 0.5 0.5 0 0.5 0 0.5 0.5 0.5 0	1.000000 0.000000 0.000000 0.000000 0.000000

Problem W. Knapsack

Time limit 2000 ms
Mem limit 262144 kB

You have a set of items, each having some integer weight not greater than 8. You denote that a subset of items is good if total weight of items in the subset does not exceed W.

You want to calculate the maximum possible weight of a good subset of items. Note that you have to consider the empty set and the original set when calculating the answer.

Input

The first line contains one integer W ($0 \le W \le 10^{18}$) — the maximum total weight of a good subset.

The second line denotes the set of items you have. It contains 8 integers cnt_1, cnt_2, ..., cnt_8 ($0 \le cnt_i \le 10^{16}$), where cnt_i is the number of items having weight i in the set.

Output

Print one integer — the maximum possible weight of a good subset of items.

Examples

Input	Output
10 1 2 3 4 5 6 7 8	10

Input	Output
0 0 0 0 0 0 0 0 0	0

Input	Output
3 0 4 1 0 0 9 8 3	3

Sailing Race

The annual sailing race is organized on a round shape lake. There are N harbors, numbered from 1 to N around the lake counterclockwise. The race consists of several stages, where each stage runs on a straight line from one harbor to another, and the race track can only meet a harbor at most once. The organizers want to create a race track that contains the highest possible number of stages. They must keep in mind that a sailboat at a given harbor may only go to some specific harbors on a route. Fortunately, for each harbor A they have the list of direct destinations from A, i.e. a list of other harbors to which a sailboat can go on a straight line from A. Generally, the race track consists of non-intersecting stages, to avoid the collision of sailboats. This year, however, a new technology is available, with which one crossing may be allowed if it lies on the first stage. So if the race track starts at harbor S and the next harbor in the track is T, then at most one stage can cross the first stage S–T. The organizers may decide to allow a crossing like this, or rather choose the classical design with non-intersecting stages.

Task

You are to write a program that computes a race track of the given type containing the highest possible number of stages.

Input

The first line of the input contains two integers, the first number (N) ($1 \le N \le 1\,000$) is the number of harbors and the second number k is the type of the required race track. If k is 0 then a classical track (without crossings) is required, while if k is 1 then the track may contain at most one crossing, as specified above. The next N lines contain the list of direct destinations from the harbors. ($i+1$)-th line contains the list for harbor i, a space-separated list of integers, terminated by 0.

Output

The first line of the output contains one integer, M, the maximal number of stages that a race track of the given type can contain. The second line contains the identifier number of the starting harbor of one such race track. If there are multiple solutions, your program should output only one; it does not matter which one.

Example input and output

input	output
7 1 5 0 5 0 7 0 3 0 4 0 4 3 0 2 1 0	5 2

Limits

Memory limit: 32 MB
Time limit: 3.0 sec

Grading

In 40% of the test cases $k = 0$. In 50% of the test cases N is at most 100.
No partial score is awarded for a correct first line.