Problem A
Ant Typing
Time Limit: 1

Consider a configurable keyboard where keys can be moved about. An ant is walking on the top row of this keyboard and needs to type a numeric string. The ant starts on the leftmost key of the top row, which contains 9 keys, some permutation of the digits from 1 to 9. On a given second, the ant can perform one of three operations:

1. Stay on that key. The digit corresponding to that key will be entered.
2. Move one key to the left. This can only happen if the ant is not on the leftmost key.
3. Move one key to the right. This can only happen if the ant is not on the rightmost key.

Compute the minimum number of seconds needed for the ant to type out the given numeric string, over all possible numeric key permutations.

Input

The single line of input contains a single string s ($1 \leq |s| \leq 10^5$) consisting only of numeric digit characters from 1 to 9. This is the numeric string that the ant needs to type.

Output

Output a single integer, which is the minimum number of seconds needed for the ant to type out the given numeric string, over all possible numeric key permutations.

Sample Input 1	Sample Output 1
78432579	20

This page is intentionally left blank.

Problem B
Bad Packing
Time Limit: 6

We have a knapsack of integral capacity and some objects of assorted integral sizes. We attempt to fill the knapsack up, but unfortunately, we are really bad at it, so we end up wasting a lot of space that can't be further filled by any of the remaining objects. In fact, we are optimally bad at this! How bad can we possibly be?

Figure out the least capacity we can use where we cannot place any of the remaining objects in the knapsack. For example, suppose we have 3 objects with weights 3, 5 and 3, and our knapsack has capacity 6. If we foolishly pack the object with weight 5 first, we cannot place either of the other two objects in the knapsack. That's the worst we can do, so 5 is the answer.

Input

The first line of input contains two integers n ($1 \leq n \leq 1{,}000$) and c ($1 \leq c \leq 10^5$), where n is the number of objects we want to pack and c is the capacity of the knapsack.

Each of the next n lines contains a single integer w ($1 \leq w \leq c$). These are the weights of the objects.

Output

Output a single integer, which is the least capacity we can use where we cannot place any of the remaining objects in the knapsack.

Sample Input 1	Sample Output 1
3 6 3 5 3	5

This page is intentionally left blank.

Problem C
Bitonic Ordering
Time Limit: 2

Noah suggests the following card game: You are given a deck of cards, each with a distinct positive integer value written on it. The cards are shuffled and placed in a row. Your objective is to arrange the cards in the row so that the values are monotonically increasing initially, and then monotonically decreasing for the remainder of the sequence.

The only move that is allowed is that two neighboring cards may swap positions. Cards may only swap positions if they are adjacent to each other.

Note that in the final ordered sequence, the initial increasing portion of the sequence may be empty (such that the whole sequence is in descending order). Likewise it is allowed for the decreasing portion of the sequence to be empty.

What is the fewest number of moves needed to get the cards arranged in the proper order?

Input

The first line of input contains a single integer n ($1 \leq n \leq 3 \cdot 10^5$), which is the number of cards.

Each of the next n lines contains a single integer c ($1 \leq c \leq 10^9$). These are the cards, in their initial order. They will all be distinct.

Output

Output a single integer, which is the fewest number of moves needed to arrange the cards as specified.

Sample Input 1	Sample Output 1
8 7 4 8 10 1 2 6 9	7

Problem D
Condorcet
Time Limit: 2

Consider an election where there are some candidates, one winner, and each voter has a complete ranked preference of the candidates. One possible method of determining the winner is to examine each pair of candidates and see which would win in a head-to-head matchup (*i.e.*, which candidate has more voters that rank them higher than their opponent) and then see if there is a single candidate that wins all their head-to-head matchups. This is called the *Condorcet Method*.

For simplicity, let `ABC` denote a vote that prefers A over B, and B over C. As an example, imagine that there are three votes: `ABC`, `BAC`, and `CAB`. Then A wins in a head-to-head matchup against B 2:1, and A also wins against C 2:1, so we could declare A the winner overall.

Note that a winner does not always exist; for example, imagine that the three votes were `ABC`, `BCA`, and `CAB` instead. Then A wins against B 2:1, B wins against C 2:1, and C wins against A 2:1. There is no single candidate that wins all their head-to-head matchups.

You are given the candidates and a set of votes. What is the minimum number of additional voters you would need to add (whose preferences you can individually control) in order to ensure that no overall winner exists? Assume that ties are broken by some tiebreaker you do not control and cannot predict. Therefore, you need to have every candidate lose a head-to-head matchup with some other candidate.

Input

The first line of input contains two integers, n ($3 \leq n \leq 5$) and m ($1 \leq m \leq n!$), where n is the number of candidates, and m is the number of vote tally lines. The candidates are represented by the first n upper-case letters of the alphabet.

Each of the next m lines contains a string s and an integer k ($1 \leq k \leq 10^6$). These are the vote tallies, which each consist of a string s defining the vote, and an integer count k indicating how many votes of type s are represented by that tally line. The string s describing a vote contains the first n upper-case letters, each exactly once, in some order. The votes in the vote tally lines are unique.

Output

Output a single integer, which is the minimum number of additional voters needed to ensure that no overall winner exists.

Sample Input 1	Sample Output 1
3 6 ABC 1 ACB 2 BAC 3 BCA 4 CAB 5 CBA 6	6

Problem E
Dominating Duos
Time Limit: 4

A group of people are standing in a line. Each person has a distinct height. You would like to count the number of unordered pairs of people in the line such that they are taller than everyone in between them in the line.

More formally, let d be a sequence of the heights of the people in order from left to right. We want to count the number of pairs of indices i and j with $i < j$ such that for all k with $i < k < j$, $d_i > d_k$ and $d_j > d_k$. Note that if $j = i + 1$ (*i.e.*, there are no k's between i and j), it is trivially true.

Input

The first line of input contains an integer n ($2 \leq n \leq 10^6$), which is the number of people.

Each of the next n lines contains a single integer d_i ($1 \leq d_i \leq n$). These are the heights of the people in the group, in the order in which they're standing. The sequence is guaranteed to be a permutation of the integers 1 through n.

Output

Output a single integer, which is the number of pairs of people who are taller than everyone between them.

Sample Input 1	Sample Output 1
3 2 1 3	3

Sample Input 2	Sample Output 2
6 1 3 2 6 4 5	7

Problem F
Exam Manipulation
Time Limit: 1

A group of students is taking a True/False exam. Each question is worth one point. You, as their teacher, want to make your students look as good as possible—so you cheat! (I know, you would never actually do that.) To cheat, you manipulate the answer key so that the lowest score in the class is as high as possible.

What is the best possible lowest score you can achieve?

Input

The first line of input contains two integers n ($1 \leq n \leq 1{,}000$) and k ($1 \leq k \leq 10$), where n is the number of students, and k is the number of True/False questions on the exam.

Each of the next n lines contains a string of length k, consisting only of upper-case '`T`' and upper-case '`F`'. This string represents the answers that a student submitted, in the order the questions were given.

Output

Output, on a single line, the best possible lowest score in the class.

Sample Input 1	Sample Output 1
5 4 TFTF TFFF TFTT TFFT TFTF	2

Sample Input 2	Sample Output 2
3 5 TFTFT TFTFT TFTFT	5

Problem G
Exciting Tournament
Time Limit: 1

A group of players compete in a no-holds-barred tournament.

Each player has a unique skill level (represented as an integer). In each game, two players play, and the player of higher skill level wins. The player of lower skill level is immediately eliminated from the tournament! The tournament continues until there is only one player left.

Due to scheduling constraints, each player has a limit on the maximum number of games they can play. Interestingly, this is the only constraint that the tournament bracket needs to meet. In other words, the bracket may not necessarily have the shape of a balanced binary tree, as long as every player plays at most their maximum number of games before getting eliminated or winning the entire tournament.

As a tournament organizer, you are free to choose any valid bracket. Given the list of participants, you wonder how exciting (or not exciting) the tournament can get. Concretely, the *excitement of a game* is defined as the bitwise XOR of the two players' skill levels. The *excitement of the tournament* is simply the sum of the *excitement* of each game.

Compute the minimum and maximum possible *excitement* values of the entire tournament.

Input

The first line of input contains a single integer n ($3 \leq n \leq 100$), which is the number of players in the tournament.

Each of the next n lines contains two integers s ($0 \leq s < 2^{30}$) and g ($2 \leq g < n$). Each line describes a single player; s is the skill level of the player, and the g is the limit on the number of games that player can play.

Output

Output two space-separated integers on a single line, which are the minimum and maximum possible *excitement* values of the entire tournament, minimum first.

Sample Input 1

```
4
41 2
13 2
36 3
17 3
```

Sample Output 1

```
94 110
```

Sample Input 2

```
6
66 5
628 4
216 5
78 4
230 5
74 3
```

Sample Output 2

```
882 2650
```

Problem H
Kth Subtree
Time Limit: 1

You are given an unrooted labeled tree. A subtree is a connected subgraph of this tree. The size of a subtree is the number of nodes in the subtree. Two subtrees are different if there is at least one node which is in one but not the other. The largest subtree is the original tree itself.

Compute the size of the K^{th} smallest non-empty subtree.

Input

The first line of input contains two integers n ($1 \leq n \leq 5{,}000$) and K ($1 \leq K \leq 10^{18}$), where n is the number of nodes in the tree, and you're looking for the size of the K^{th} smallest subtree. The nodes are numbered 1 through n.

Each of the next $n-1$ lines contains a pair of integers u and v ($1 \leq u, v \leq n, u \neq v$), which represents an undirected edge between nodes u and v. All edges are distinct. It is guaranteed that the edges form a single tree.

Output

Output a single integer, which is the number of nodes in the K^{th} smallest non-empty subtree of the input tree. If there are fewer than K non-empty subtrees of the given tree, output -1.

Sample Input 1	Sample Output 1
2 1 1 2	1

Sample Input 2	Sample Output 2
2 3 1 2	2

Sample Input 3	Sample Output 3
5 10 1 2 2 3 3 4 4 5	3

Problem I
Longest Common Subsequence
Time Limit: 1

You are given n strings, each a permutation of the first k upper-case letters of the alphabet.

String s is a *subsequence* of string t if and only if it is possible to delete some (possibly zero) characters from the string t to get the string s.

Compute the length of the longest common *subsequence* of all n strings.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq 26$), where n is the number of strings, and the strings are all permutations of the first k upper-case letters of the alphabet.

Each of the next n lines contains a single string t. It is guaranteed that every t contains each of the first k upper-case letters of the alphabet exactly once.

Output

Output a single integer, the length of the longest subsequence that appears in all n strings.

Sample Input 1	Sample Output 1
2 3 BAC ABC	2

Sample Input 2	Sample Output 2
3 8 HGBDFCAE ADBGHFCE HCFGBDAE	3

Sample Input 3	Sample Output 3
6 8 AHFBGDCE FABGCEHD AHDGFBCE DABHGCFE ABCHFEDG DGABHFCE	4

Problem J
Magic Trick
Time Limit: 1

You are performing a magic trick with a special deck of cards.

You lay out the cards in a row from left to right, face up. Each card has a lower-case letter on it. Two cards with the same letter are indistinguishable. You select an audience member to perform an operation on the cards. You will not see what operation they perform.

The audience member can do one of two things—they can either select any two cards and swap them, or leave the cards untouched.

In order for the trick to succeed, you must correctly guess what the audience member did—either you guess that the audience member did nothing, or you point at the two cards the audience member swapped.

Given a string that represents the initial arrangement of the cards, can you guarantee that you will always be able to guess the audience member's operation correctly, no matter what operation they perform?

Input

The input consists of a single line containing the string s ($1 \leq |s| \leq 50$), which represents the initial arrangement of the cards, in the order they appear in the row. The string contains only lower-case letters ('a'–'z').

Output

Output a single line with 1 if you can guarantee that you will always be able to guess the audience member's operation correctly, or 0 otherwise.

Sample Input 1	Sample Output 1
robust	1

Sample Input 2	Sample Output 2
icpc	0

This page is intentionally left blank.

Problem K
Missing Number
Time Limit: 1

You are teaching kindergarten! You wrote down the numbers from 1 to n, in order, on a whiteboard. When you weren't paying attention, one of your students erased one of the numbers.

Can you tell which number your mischievous student erased?

Input

The first line of input contains a single integer n ($2 \leq n \leq 100$), which is the number of numbers that you wrote down.

The second line of input contains a string of digits, which represents the numbers you wrote down (minus the one that has been erased). There are no spaces in this string. It is guaranteed to contain all of the numbers from 1 to n, in order, except for the single number that the student erased.

Output

Output a single integer, which is the number that the tricky student erased.

Sample Input 1	Sample Output 1
5 1235	4

Sample Input 2	Sample Output 2
10 1234568910	7

Sample Input 3	Sample Output 3
15 1234567891012131415	11

This page is intentionally left blank.

Problem L
Rainbow Numbers
Time Limit: 1

Define a *rainbow number* as an integer that, when represented in base 10 with no leading zeros, has no two adjacent digits the same.

Given lower and upper bounds, count the number of rainbow numbers between them (inclusive).

Input

The first line of input contains a single integer L ($1 \leq L < 10^{10^5}$), which is the lower bound.

The second line of input contains a single integer U ($1 \leq U < 10^{10^5}$), which is the upper bound.

It is guaranteed that $L \leq U$. Note that the limits are not a misprint; L and U can be up to 10^5 digits long.

Output

Output a single integer, which is the number of rainbow numbers between L and U (inclusive). Because this number may be very large, output it modulo $998{,}244{,}353$.

Sample Input 1	Sample Output 1
1 10	10

Sample Input 2	Sample Output 2
12345 65432	35882

This page is intentionally left blank.

Problem M
Rating Problems
Time Limit: 1

Your judges are preparing a problem set, and they're trying to evaluate a problem for inclusion in the set. Each judge rates the problem with an integer between -3 and 3, where:

3	means: I *really* like this problem!
-3	means: I *really don't* like this problem!
0	means: Meh. I don't care if we use this problem or not.

The overall rating of the problem is the average of all of the judges' ratings—that is, the sum of the ratings divided by the number of judges providing a rating.

Some judges have already rated the problem. Compute the minimum and maximum possible overall rating that the problem can end up with after the other judges submit their ratings.

Input

The first line of input contains two integers n ($1 \leq n \leq 10$) and k ($0 \leq k \leq n$), where n is the total number of judges, and k is the number of judges who have already rated the problem.

Each of the next k lines contains a single integer r ($-3 \leq r \leq 3$). These are the ratings of the k judges that have already rated the problem.

Output

Output two space-separated floating point numbers on a single line, which are the minimum and maximum overall rating the problem could achieve after the remaining judges rate the problem, minimum first. These values must be accurate to an absolute or relative error of 10^{-4}.

Sample Input 1	Sample Output 1
5 2 1 2	-1.2 2.4

Sample Input 2	Sample Output 2
4 4 -3 -3 -2 -3	-2.75 -2.75

Problem N
Triangular Collection
Time Limit: 1

Call a set of positive integers *triangular* if it has size at least three and, for all triples of distinct integers from the set, a triangle with those three integers as side lengths can be constructed.

Given a set of positive integers, compute the number of its *triangular* subsets.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50$), which is the number of integers in the set.

Each of the the next n lines contains a single integer x ($1 \leq x \leq 10^9$). These are the elements of the set. They are guaranteed to be distinct.

Output

Output a single integer, which is the number of triangular subsets of the given set.

Sample Input 1	Sample Output 1
5 3 1 5 9 10	2

Sample Input 2	Sample Output 2
10 27 26 17 10 2 14 1 12 23 39	58

Problem O
TripTik
Time Limit: 9

Have you ever been on a long road journey? AAA (the American Automobile Association) has a tool for long road trips. It's called a TripTik, and it follows the highways, showing points of interest.

You are building a TripTik app, which allows users to see what's on their route. It models a highway as a straight line, and points of interest as points along that line. All points have an integer coordinate as well as a unique integral weight. Your app provides a viewport, which can scale in and out. Also, to prevent the display from becoming too cluttered, only a small number of the points with the highest weights are shown. The initial viewport is centered at 0.0, and shows from -1.0 to 1.0 on the line.

There are three valid operations for changing your viewport:

1. ***Zoom out***: double the dimensions of your viewport while keeping the center the same; this can *always* be done regardless of the current dimensions of the viewport.
2. ***Zoom in***: halve the dimensions of your viewport while keeping the center the same; this can *always* be done regardless of the current dimensions of the viewport.
3. ***Recenter***: change the center of your viewport to be equal to a point of interest visible in your viewport (including the boundary).

There is an important caveat: Your TripTik app will not render all points of interest in a given viewport; instead, it will only render a certain number of points in the viewport with the highest weights. The remaining points with lower weight are not visible, and therefore are not valid targets for the recenter operation.

For each point of interest, determine the minimum number of operations needed to go from the starting viewport to a viewport where that point of interest is centered and visible. Consider each point of interest independently.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq 4$), where n is the number of points, and k is the maximum number of points visible in the viewport.

Each of the next n lines contains a single integer x ($|x| \leq 10^8, x \neq 0$). These are the points of interest. The weight of each point is equal to its position in the list, with lower weights earlier in the list. All points will be distinct.

Output

Output n lines, each with a single integer, indicating the minimum number of operations necessary to get from the starting viewport to a viewport which is centered on the corresponding input point and can see that point. Output -1 if this isn't possible. The order of output lines should correspond to the order of inputs for which they're the answer.

Sample Input 1	Sample Output 1
4 2 100 4 1 3	-1 5 1 3

Problem P
Unread Messages
Time Limit: 1

There is a group of people in an internet email message group. Messages are sent to all members of the group, and no two messages are sent at the same time.

Immediately before a person sends a message, they read all their unread messages up to that point. Each sender also reads their own message the moment it is sent. Therefore, a person's unread messages are exactly the set of messages sent after that person's last message.

Each time a message is sent, compute the total number of unread messages over all group members.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^9$) and m ($1 \leq m \leq 1{,}000$), where n is the number of people in the group, and m is the number of messages sent. The group members are identified by number, 1 through n.

Each of the next m lines contains a single integer s ($1 \leq s \leq n$), which is the sender of that message. These lines are in chronological order.

Output

Output m lines, each with a single integer, indicating the total number of unread messages over all group members, immediately after each message is sent.

Sample Input 1	Sample Output 1
2 4 1 2 1 2	1 1 1 1

Sample Input 2	Sample Output 2
3 9 1 2 3 2 1 3 3 2 1	2 3 3 4 3 3 5 4 3