Problem A. Arrangement of Contest

Source file name:	arrange.c, arrange.cpp, arrange.java
Input:	Standard
Output:	Standard

Little Dmitry and little Petr want to arrange a contest. Their little friends submitted several task proposals and now Dmitry and Petr want to select some of them for the contest. As they are just little boys, they cannot estimate quality of tasks, but they know for sure that in *good* contest title of the first problem starts with A, the title of the second one — with B, and so on.

Given titles of the proposed tasks, help little brothers to determine the maximal number of problems in a *good* contest they can arrange.

Input

The first line contains single integer n — the number of problem proposals received by the little brothers ($1 \le n \le 100$).

Next n lines contain titles of proposed problems, one per line. The length of each title does not exceed 30 characters. Each title starts with an uppercase letter and contains only English letters, digits and underscores.

Output

Output a single number — the maximal number of problems in a *good* contest. In case there is no *good* contest that may be arranged, output 0.

Example

Input	Output
12 Arrangement_of_Contest Ballot_Analyzing_Device Correcting_Curiosity Dwarf_Tower Energy_Tycoon Flight_Boarding_Optimization Garage Heavy_Chain_Clusterization Intellectual_Property J Kids_in_a_Friendly_Class Lonely_Mountain	12
3 Snow_White_and_the_7_Dwarfs A_Problem Another_Problem	1
2 Good_Problem Better_Problem	0

Problem B. Ballot Analyzing Device

Source file name:	bad.c, bad.cpp, bad.java
Input:	`Standard`
Output:	`Standard`

Election committee of Flatland is preparing for presidential elections. To minimize human factor in ballot counting they decided to develop an automated Ballot Analyzing Device (BAD).

There are n candidates running for president. The ballot contains one square field for each candidate. The voter must mark exactly one of the fields. If no field is marked or there are two or more marked fields, the ballot is invalid. Each voter puts his/her ballot to a special scanner in BAD. The scanner analyzes marks on the ballot and creates a special *voting string* of n characters: 'X' for marked field and '.' for unmarked one. Now voting strings must be analyzed to get the report. Your task is to develop a report generator for BAD.

Given voting strings for all ballots, your program must print the voting report. Candidates in the report must be arranged in order of decreasing number of votes. If two candidates have the same number of votes, they must have the same order as in a voting ballot. For each candidate calculate his/her result in percent (if the candidate received p votes, the result in percent is $100p/m$). The last line of the report must indicate the percentage of the invalid ballots.

Input

The first line contains two integers n and m — the number of candidates and the number of ballots ($2 \le n \le 10$; $1 \le m \le 1000$). The following n lines contain last names of candidates. Each name is a string of at most 100 English letters. There is no candidate named **"Invalid"**.

Then m lines follow, each of them contains one voting string.

Output

Print $n+1$ lines. First print results for candidates in percent. For each candidate print his/her last name followed by a space and then his/her result in percent and a percent sign '%'. The last line must specify the percentage of invalid ballots: a word **"Invalid"** followed by a space, the percentage of invalid ballots and a percent sign.

Round all numbers to exactly two digits after the decimal point. If the number is exactly in the middle of two representable numbers, output the greater one (e.g. output "12.35" for 12.345).

Example

Input	Output
4 7 Loudy Apples Dogman Miller .X.. X... .... ..X. ..XX ..X. ..X.	Dogman 42.86% Loudy 14.29% Apples 14.29% Miller 0.00% Invalid 28.57%

Problem C. Trener

Source file name:	trener.c, trener.cpp, trener.java
Input:	`Standard`
Output:	`Standard`

Mirko has been moving up in the world of basketball, starting as a mere spectator, mastering snack salesmanship, finally reach the coveted position of the national team coach. He is now facing a difficult task: selecting the five primary players for the upcoming match against Tajikistan.

Since Mirko is incredibly lazy, he doesn't bother remembering players' names, let alone their actual skills. That's why he has settled on selecting five players who share the same first letter of their surnames, so that he can remember them more easily. If there are no five players sharing the first letter of their surnames, Mirko will simply forfeit the game!

In order to obtain insight into possibilities for his team, Mirko wants to know all the different letters that his primary team's surnames may begin with.

Input

The first line of input contains the positive integer **N** ($1 \leq \mathbf{N} \leq 150$), the number of players that Mirko has available.

Each of the following **N** lines contains one word (at most 30 characters long, consisting only of lowercase English letters), a surname of one of the players.

Output

If there are no five players that Mirko can select matching his criteria, output a single line containing the word "PREDAJA" (without quotes). Otherwise, output all possible first letters of representation player surnames, sorted lexicographically, in a single line with no spaces.

Example

Input	Output
18 babic keksic boric bukic sarmic balic kruzic hrenovkic beslic boksic krafnic pecivic klavirkovic kukumaric sunkic kolacic kovacic prijestolonasljednikovic	bk
6 michael jordan lebron james kobe bryant	PREDAJA

Explication

Clarification of the first example: Mirko can choose between teams with all players' surnames begining with either 'k' or 'b'.

Problem D. Dwarf Tower

Source file name:	dwarf.c, dwarf.cpp, dwarf.java
Input:	`Standard`
Output:	`Standard`

Little Vasya is playing a new game named "Dwarf Tower". In this game there are n different items, which you can put on your dwarf character. Items are numbered from 1 to n. Vasya wants to get the item with number 1.

There are two ways to obtain an item:

- You can buy an item. The i-th item costs c_i money.
- You can craft an item. This game supports only m types of crafting. To craft an item, you give two particular different items and get another one as a result.

Help Vasya to spend the least amount of money to get the item number 1.

Input

The first line of input contains two integers n and m ($1 \le n \le 10\,000; 0 \le m \le 100\,000$) — the number of different items and the number of crafting types.

The second line contains n integers c_i — values of the items ($0 \le c_i \le 10^9$).

The following m lines describe crafting types, each line contains three distinct integers a_i, x_i, y_i — a_i is the item that can be crafted from items x_i and y_i ($1 \le a_i, x_i, y_i \le n$; $a_i \ne x_i$; $x_i \ne y_i$; $y_i \ne a_i$).

Output

The output should contain a single integer — the least amount of money to spend.

Example

Input	Output
5 3 5 0 1 2 5 5 2 3 4 2 3 1 4 5	2
3 1 2 2 1 1 2 3	2

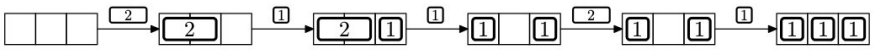
Problem E. Energy Tycoon

Source file name:	energy.c, energy.cpp, energy.java
Input:	`Standard`
Output:	`Standard`

Little Vasya is playing a new computer game — turn-based strategy "Energy Tycoon".

The rules of the game are quite simple:

- The board contains n slots arranged in a line.
- There are power plants, one power plant occupies one or two consecutive slots, and produces one unit of energy.
- Each turn the game allows you to build one new power plant, you can put it on the board if you wish. If there is no place for the new power plant, you can remove some older power plants.
- After each turn, the computer counts the amount of energy produced by the power plants on the board and adds it to the total score.



Vasya already knows the types of power plant he will be able to build each turn. Now he wants to know, what the maximum possible score he can get is. Can you help him?

Input

The first line of the input contains one integer n ($1 \le n \le 100\,000$) — the number of slots on the board. The second line contains the string s. The i-th character of the string is 1 if you can build one-slot power plant at the i-th turn and the character is 2 if you can build two-slot power plant at the i-th turn. The number of turns does not exceed 100 000.

Output

The output should contain a single integer — the maximal score that can be achieved.

Example

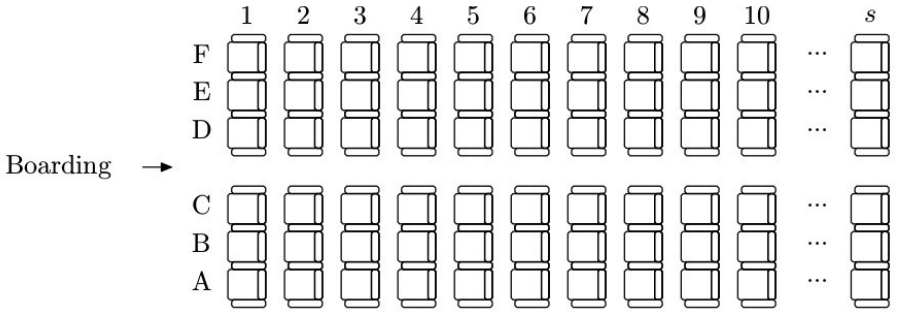
Input	Output
3 21121	10
2 12	2
2 211	4

The picture shows the optimal sequence of moves for the first example.

Problem F. Flight Boarding Optimization

Source file name:	flight.c, flight.cpp, flight.java
Input:	`Standard`
Output:	`Standard`

Peter is an executive boarding manager in Byteland airport. His job is to optimize the boarding process. The planes in Byteland have s rows, numbered from 1 to s. Every row has six seats, labeled A to F.



There are n passengers, they form a queue and board the plane one by one. If the i-th passenger sits in a row r_i then the difficulty of boarding for him is equal to the number of passengers boarded before him and sit in rows $1 \dots r_i - 1$. The total difficulty of the boarding is the sum of difficulties for all passengers. For example, if there are ten passengers, and their seats are 6A, 4B, 2E, 5F, 2A, 3F, 1C, 10E, 8B, 5A, in the queue order, then the difficulties of their boarding are 0, 0, 0, 2, 0, 2, 0, 7, 7, 5, and the total difficulty is 23.

To optimize the boarding, Peter wants to divide the plane into k zones. Every zone must be a continuous range of rows. Than the boarding process is performed in k phases. On every phase, one zone is selected and passengers whose seats are in this zone are boarding in the order they were in the initial queue.

In the example above, if we divide the plane into two zones: rows 5–10 and rows 1–4, then during the first phase the passengers will take seats 6A, 5F, 10E, 8B, 5A, and during the second phase the passengers will take seats 4B, 2E, 2A, 3F, 1C, in this order. The total difficulty of the boarding will be 6.

Help Peter to find the division of the plane into k zones which minimizes the total difficulty of the boarding, given a specific queue of passengers.

Input

The first line contains three integers n ($1 \le n \le 1000$), s ($1 \le s \le 1000$), and k ($1 \le k \le 50$; $k \le s$). The next line contains n integers r_i ($1 \le r_i \le s$).

Each row is occupied by at most 6 passengers.

Output

Output one number, the minimal possible difficulty of the boarding.

Example

Input	Output
10 12 2 6 4 2 5 2 3 1 11 8 5	6

Problem G. Garage

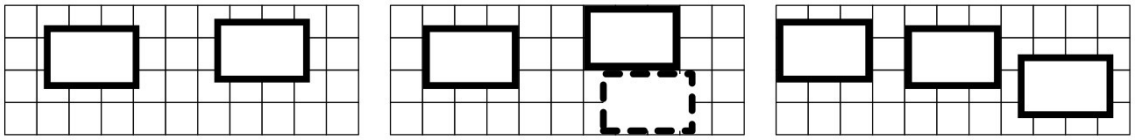
Source file name:	garage.c, garage.cpp, garage.java
Input:	`Standard`
Output:	`Standard`

Wow! What a lucky day! Your company has just won a social contract for building a garage complex. Almost all formalities are done: contract payment is already transferred to your account.

So now it is the right time to read the contract. Okay, there is a sandlot in the form of $W \times H$ rectangle and you have to place some garages there. Garages are $w \times h$ rectangles and their edges must be parallel to the corresponding edges of the sandlot (you may not rotate garages, even by 90°). The coordinates of garages may be non-integer.

You know that the economy must be economical, so you decided to place as *few* garages as possible. Unfortunately, there is an opposite requirement in the contract: placing maximum possible number of garages.

Now let's see how these requirements are checked... The plan is accepted if it is impossible to add a new garage without moving the other garages (the new garage must also have edges parallel to corresponding sandlot edges).



Accepted optimal plan | Rejected plan | Accepted, but non-optimal plan

Time is money, find the minimal number of garages that must be ordered, so that you can place them on the sandlot and there is no place for an extra garage.

Input

The only line contains four integers: W, H, w, h — dimensions of sandlot and garage in meters. You may assume that $1 \le w \le W \le 30\,000$ and $1 \le h \le H \le 30\,000$.

Output

Output the optimal number of garages.

Example

Input	Output
11 4 3 2	2
10 8 3 4	2
15 7 4 2	4

The plan on the first picture is accepted and optimal for the first example. Note that a rotated (2×3) garage could be placed on the sandlot, but it is prohibited by the contract.

Problem H. Kusac

Source file name:	kusac.c, kusac.cpp, kusac.java
Input:	`Standard`
Output:	`Standard`

Mirko has given up on the difficult coach job and switched to food tasting instead. Having skipped breakfast like a professional connoisseur, he is visiting a Croatian cured meat festival. The most renowned cook at the festival, Marijan Bajs, has prepared **N** equal sausages which need to be distributed to **M** tasters such that each taster gets a precisely equal amount. He will use his trusted knife to cut them into pieces.

In order to elegantly divide the sausages, the **number of cuts** splitting individual sausages must be **as small as possible.** For instance, if there are two sausages and six tasters (the first test case below), it is sufficient to split each sausage into three equal parts, making a total of four cuts. On the other hand, if there are three sausages and four tasters (the second test case below), one possibility is cutting off three quarters of each sausage. Those larger parts will each go to one of the tasrers, while the fourth taster will get the three smaller pieces (quarters) left over.

Mirko wants to try the famous sausages, so he volunteered to help Bajs. Help them calculate the minimum total number of cuts needed to carry out the desired division.

Input

The first and only line of input contains two positive integers, **N** and **M** ($1 \leq$**N**, **M** ≤ 100), the number of sausages and tasters, respectively.

Output

The first and only line of output must contain the required minimum number of cuts.

Example

Input	Output
2 6	4
3 4	3
6 2	0

Problem I. Ratar

Source file name:	ratar.c, ratar.cpp, ratar.java
Input:	`Standard`
Output:	`Standard`

After Mirko's failed stint as a coach and a passing obsession with Croatian meat delicacies, his weight problems have motivated him to work hard as a farmer. He has moved to a village where his friend Slavko lives. Farmers in the village share a large common plot of land in the shape of a **NxN** square, divided into $\mathbf{N^2}$ unit squares. A unit square at coordinates $(\mathbf{i}, \mathbf{j})$ brings in the income of $\mathbf{A_{ij}}$, which can be negative (for example, if the square has to be maintained but is not cultivated). The farmers always divide the common land into smaller **rectangular fields** with edges **parallel** to the common land edges.

Slavko is skeptical of Mirko since his failure as a coach, so he insists that both of them are assigned land with the **same total income**, but also thet the two plots share **exactly one** common corner so that the two friends can keep an eye on each other (Slavko knows that Mirko is prone to mischief). The common corner must be the only point where the two plots meet, in order to prevent border-related arguments.

You are given a description of the common land plot. Find the total number of plot pairs that satisfy Slavko's criteria.

Input

The first line of input contains the positive integer $\mathbf{N}$ ($1 \leq \mathbf{N} \leq 50$), the dimensions of the common land plot. Each of the following $\mathbf{N}$ lines contains $\mathbf{N}$ space-separated numbers $\mathbf{A_{ij}}$ ($-1000 < \mathbf{A_{ij}} < 1000$), the income provided by the respective cell.

Output

The first and only line of output must contain the total number of plot pairs satisfying the given condition.

Example

Input	Output
3 1 2 3 2 3 4 3 4 8	7
4 -1 -1 -1 -1 1 2 3 4 1 2 3 4 1 2 3 4	10
5 -1 -1 -1 -1 -1 -2 -2 -2 -2 -2 -3 -3 -3 -3 -3 -4 -4 -4 -4 -4 -5 -5 -5 -5 -5	36

Explication

Clarification of the first example: The possible rectangle pairs are: (0,0)-(1,1) and (2,2)-(2,2), (1,0)-(1,0) and (0,1)-(0,1), (2,0)-(2,0) and (1,1)-(1,1), (1,1)-(1,1) and (0,2)-(0,2), (2,1)-(2,1) and (1,2)-(1,2), (2,0)-(2,1) and (0,2)-(1,2), (1,0)-(2,0) and (0,1)-(0,2).

Problem J. J

Source file name:	j.c, j.cpp, j.java
Input:	`Standard`
Output:	`Standard`

> The J programming language, developed in the early 1990s by Kenneth E. Iverson and Roger Hui, is a synthesis of APL (also by Iverson) and the FP and FL function-level languages created by John Backus.
>
> Wikipedia. J (programming language)

APL language family is famous for its support of advanced operations on vectors and arrays, and J is not an exception. For example, all unary and binary numeric operations in these languages by default are applicable to vectors and arrays of different dimensions. Plus operation ('+') can add not only scalars, like in other languages, but also scalars and vectors (the scalar is added to each component of the vector), or vectors and vectors (the vectors are added component-wise).

The expressive power of J is amazing (as well as its cryptic syntax), but for this problem we need just a small subset of the language. We consider a single expression, where we may use one vector variable X, one scalar variable N — the length of the vector X, and the following operations:

- We can add ('+'), subtract ('-') or multiply ('`*`') two vectors, vector and scalar, or two scalars.
- We can use unary minus ('-') and unary squaring operations ('`*:`') for scalars and vectors (component-wise).
- We can *fold* a vector with plus operation ('+/') — that is, compute the sum of a vector (unary operation).

Operations are evaluated right-to-left, natural precedence of operations is ignored in J. The order of evaluation can be altered by parentheses. More precisely the syntax is specified in the following BNF.

⟨*expression*⟩ ::= ⟨*term*⟩ | ⟨*term*⟩ ('`+`'|'`-`'|'`*`') ⟨*expression*⟩ | ('`-`'|'`*:`'|'`+/`') ⟨*expression*⟩

⟨*term*⟩ ::= '`(`' ⟨*expression*⟩ '`)`' | '`X`' | '`N`' | ⟨*number*⟩

⟨*number*⟩ ::= ('`0`'|'`1`'|...|'`9`')$^+$

To correctly impose one more limitation on expression syntax, let us define *complexity* of an expression:

- complexity of scalars (numbers, 'N', and result of fold) is zero;
- complexity of 'X' is one;
- complexity of addition and subtraction is the maximum of their operands' complexities;
- complexity of multiplication is the sum of its operands' complexities;
- complexity of unary squaring is twice the complexity of its operand.

For example, the complexity of expression ''`(3-+/*:*:X)-X**:X`'' is 3, while the complexity of its subexpression ''`*:*:X`'' is 4.

Your program is given a scalar-valued expression and a value of the vector X, and it should compute the expression result modulo 10^9. The complexity of each subexpression in the given expression does not exceed 10.

Input

The first line contains one integer number N ($1 \le N \le 10^5$) — the length of the vector X.

The second line contains N integers — components of the vector X ($0 \le X_i < 10^9$).

The third line contains the expression to be computed, a non-empty string of not more than 10^5 symbols. Each number in the expression is less than 10^9. The fold is never applied to a scalar.

Output

Output a single integer number — the expression result modulo 10^9.

Example

Input	Output
5 1 2 3 4 5 +/*:X	55
5 1 2 3 4 5 N++/X-X+1	0
3 11 56 37 +/(3-+/*:*:X)-X**:X	964602515

The first expression computes squared length of the vector X in Euclidean metrics.

The second expression result does not depend on X and always equals to zero.

The actual result of the third expression is -35397485.

Problem K. Lopov

Source file name:	lopov.c, lopov.cpp, lopov.java
Input:	`Standard`
Output:	`Standard`

The difficult economic situation in the country and reductions in government agricultural subsidy funding have caused Mirko to change his career again, this time to a thief. His first professional endeavour is a jewellery store heist.

The store contains **N** pieces of jewellery, and each piece has some mass $\mathbf{M_i}$ and value $\mathbf{V_i}$. Mirko has **K** bags to store his loot, and each bag can hold some maximum mass $\mathbf{C_i}$. He plans to store all his loot in these bags, but **at most one** jewellery piece in each bag, in order to reduce the likelihood of damage during the escape.

Find the maximum total jewellery value that Mirko can "liberate".

Input

The first line of input contains two numbers, **N** and $\mathbf{K}(1 \leq N, K \leq 300000)$. Each of the following **N** lines contains a pair of numbers, $\mathbf{M_i}$ and $\mathbf{V_i}$ $(1\mathbf{M_i}, \mathbf{V_i} \leq 1000000)$. Each of the following **K** lines contains a number, $\mathbf{C_i}(1 \leq \mathbf{C_i} \leq 100000000)$. All numbers in the input are positive integers.

Output

The first and only line of output must contain the maximum possible total jewellery value.

Example

Input	Output
2 1 5 10 100 100 11	10
3 2 1 65 5 23 2 99 10 2	164

Explication

Clarification of the second example: Mirko stores the first piece of jewellery into the second bag and the third piece into the first bag.

Problem L. Organizator

Source file name:	organizator.c, organizator.cpp, organizator.java
Input:	`Standard`
Output:	`Standard`

Unexpected problems with law enforcement have convinced Mirko to take up a less lucrative but less morally ambiguous career: he has become the chief organizer of a team computer science contest.

There are **N** CS clubs that wish to participate in the contest. The presidents of the clubs are quite stubborn and will participate in the contest **only if** the contest team size makes it possible for all club members to participate.

The contest consists of two rounds: **qualifications** and **finals**. All teams that are competing must have an **equal** number of members and **all members** of one team must belong to the same club. Any number of teams from each club can participate in the qualification round, and the **best team** from each club earns a spot in the **finals**.

Mirko is aware that, with a new and unproven contest, he needs publicity. For that reason, he wants to set the team size such that **the number of individual participants** in the **finals** is as **large** as possible. Remember, each club that participates has a right to **one** team in the **finals**. Furthermore, at least **two** clubs must participate in the contest, otherwise the contest would be too boring to attract sponsors.

Determine the maximum possible number of participants in the finals so that Mirko can double check his team size choice.

Input

The first line of input contains the positive integer **N** ($2 \leq \mathbf{N} \leq 200000$), the number of clubs. The second line of input contains **N** space-separated integers from the interval [1, 2000000], the number of members of each club.

Output

The first and only line of output must contain the maximum possible number of finalists.

Example

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

Explication

Clarification of the first example: Mirko decides on 2 members per team, so clubs 2 and 3 participate.