The 3rd Universal Cup

Stage 24: Poland

January 4-5, 2025

This problem set should contain 12 problems (A to L) on 19 numbered pages.

Based on

Akademickie Mistrzostwa Polski w Programowaniu Zespoowym (AMPPZ)

Hosted by

Problem A. Acronym

Time limit: 2 seconds
Memory limit: 1024 megabytes

Aga loves secrets and word games. Tomorrow is her name day (*imieniny* in Polish). Her friend Ania is decorating a greeting card but would like to include a sentence (a sequence of words) whose first letters form a hidden code. Such a concatenation (joining) of the first letters is called an acronym.

Ania has already come up with the following example sentences for the occasion:

- `WONDERFUL IMMENSE SUPERB HAPPINESS` with the acronym `WISH`.
- `LIVE LAUGH LOVE` with the acronym `LLL`.
- `AKRONIM NA IMIENINY AGI` with the acronym `ANIA`.
- `BABY AND BABY YETI` with the acronym `BABY`.

The last example shows that words in the sentence can repeat; we can also use the acronym within a sentence.

Ania has to decide on an acronym, so she asked for your help. You are given a dictionary, i.e. a set of n different permitted words. Check if it is possible to construct a sentence from these words such that the first letters of each word in the sentence form a word also in the dictionary. Provide an example of such a sentence or output `-1` if it is impossible.

Input

The first line contains an integer n ($1 \le n \le 200$), denoting the number of words in the dictionary.

Each of the following n lines contains one word s_i consisting of at least 2 and at most 8 uppercase English alphabet letters `A-Z`. The words in the input are pairwise distinct.

Output

If no such sentence exists, output the number `-1` in a single line.

Otherwise, in the first line, output the number of words in the found sentence, and in the second line, output the sentence as a sequence of words separated by spaces.

If multiple sentences meet the criteria, output any one of them.

Examples

standard input	standard output
5 ANIA IMIENINY AGI AKRONIM NA	4 AKRONIM NA IMIENINY AGI
1 XX	2 XX XX

Note

In the first example test many other sentences are also valid, for example: `NA ANIA` or `ANIA NA IMIENINY ANIA`. However, sentences like `AKRONIM NA IMIENINY ANI` and `AKRONIM NA IMIENINY` are incorrect since the word `ANI` is not in the dictionary.

In the second example test the sentence `XX XX` is the only correct answer.

Problem B. Baggage

Time limit:	3 seconds
Memory limit:	1024 megabytes

As an avid traveller, Bartek sets out each year to explore new countries and continents. He manages this by carefully optimizing his travel costs – a must when travelling with two suitcases. To save money, he relies on his friends to store his suitcases and stays in hotels paid for by organizers of local programming competitions.

There are n cities in the world (numbered from 1 to n) and m flight connections. Each flight connection is described by four numbers (a, b, c, d), representing a one-way flight from city a to city b with a cost of c and a limit of d suitcases. Bartek can use the same flight multiple times, each time bringing along at most d suitcases. The ticket price does not depend on the amount of luggage. A suitcase cannot travel alone on the plane, but in each city Bartek has a friend who might store any number of suitcases for any length of time and as many times as needed.

Bartek has two suitcases and wants to travel between two cities. For each of n^2 pairs of cities (x, y) find the minimum total travel cost if Bartek starts in city x with two suitcases and wants to arrive with them in city y. Output `-1` if it isn't possible for a given pair of cities.

Input

The first line contains two integers n and m ($1 \le n \le 400$; $0 \le m \le 500\,000$), denoting the number of cities and the number of flight connections.

Each of the following m lines contains four integers a_i, b_i, c_i, d_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$; $1 \le c_i \le 10^9$; $0 \le d_i \le 2$) describing one flight connection. There might be multiple connections from one city to another. It is possible that some cities have no outgoing or incoming connections.

Output

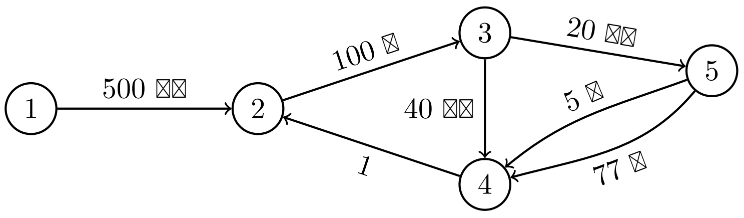
Output n lines, each containing n integers separated by spaces. In the i-th line, the j-th integer should be the minimum total cost of travelling with two suitcases from city i to city j, or integer `-1` if it is not possible.

Example

standard input	standard output

`5 7`
`1 2 500 2`
`2 3 100 1`
`3 5 20 2`
`5 4 5 1`
`4 2 1 0`
`3 4 40 2`
`5 4 77 1`
	`0 500 726 751 746`
	`-1 0 226 251 246`
	`-1 -1 0 40 20`
	`-1 -1 -1 0 -1`
	`-1 -1 -1 131 0`

Note



For example, from city 1 to city 3 Bartek can travel in the following way:

$1 \to 2$ (drop a suitcase) $\to 3$ (drop a suitcase) $\to 5 \to 4 \to 2$ (pick up a suitcase) $\to 3$ (pick up a suitcase)

The total cost is $500 + 100 + 20 + 5 + 1 + 100 = 726$.

Problem C. Cows

Time limit:	4 seconds
Memory limit:	1024 megabytes

A long pasture is divided into n segments numbered consecutively from 1 to n. The initial height of the grass in the i-th segment is h_i. There is one cow grazing[1] in each segment. The segments are separated by fences. The cows cannot move between segments, but they can stretch their heads over the fence to eat grass from an adjacent segment (i and $i+1$ are adjacent for every i from 1 to $n-1$).

Each minute, each cow eats grass from one chosen segment:

- If there is still grass in her own segment ($h_i > 0$), the cow always chooses her own segment.
- Otherwise, the cow chooses one of the adjacent segments that still has grass.
- The cow does nothing if there is no grass in her segment and the adjacent segments.

If x cows eat grass from the same segment, they reduce it by x in one minute; though the grass height cannot fall below zero. So after a minute we have $h_i := \max(0, h_i - x)$.

The cows cooperate to eat all the grass from the pasture as quickly as possible. After how many minutes can they succeed?

Input

The first line contains an integer n ($1 \le n \le 200\,000$) denoting the number of segments in the pasture.

The second line contains n integers $h_1, h_2, \ldots, h_n$ ($0 \le h_i \le 10^9$) denoting the initial heights of grass in the consecutive segments. At least one of the values h_i is positive.

Output

Output one integer – the minimum time required to eat all the grass, in minutes.

Examples

standard input	standard output
5 5 4 0 4 6	4
3 1 4 6	5

Note

In the first example test the optimal strategy is as follows:

- $[5, 4, 0, 4, 6]$ – Cow 3 chooses the adjacent segment 4. The other cows eat in their own segments.
- $[4, 3, 0, 2, 5]$ – Cow 3 chooses segment 4.
- $[3, 2, 0, 0, 4]$ – Cow 3 chooses segment 2, and cow 4 chooses segment 5.
- $[2, 0, 0, 0, 2]$ – Cow 3 does nothing; cow 2 chooses segment 1; cow 4 chooses segment 5.
- $[0, 0, 0, 0, 0]$ – The cows have eaten all the grass in 4 minutes.

In the second example test the cows never have a choice. The process must proceed as follows:

$$[1, 4, 6] \to [0, 3, 5] \to [0, 1, 4] \to [0, 0, 3] \to [0, 0, 1] \to [0, 0, 0]$$

[1] *To graze* is a verb used for animals eating plants, e.g. cows and sheep eating grass in a field.

Problem D. Diminishing Fractions

Time limit:	2 seconds
Memory limit:	1024 megabytes

Daria has implemented a program for rational number arithmetic. The program can calculate the result of an expression consisting of addition and subtraction of at most n fractions, where the numerators and denominators are positive integers not greater than n.

Daria is concerned about the program's efficiency when the result is either a very large number or a positive number very close to zero. She has already tested the first case easily by entering the expression $\frac{n}{1} + \frac{n}{1} + \ldots + \frac{n}{1}$. But what about obtaining a very small positive number?

You are given t test cases, each with a limit n considered by Daria. For each given n find an expression whose value is the smallest possible positive number.

Input

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

Each of the following t lines contains an integer n ($1 \leq n \leq 50\,000$).

The sum of all values of n does not exceed $4 \cdot 10^6 = 4\,000\,000$.

Output

For each test case, output one line with the found expression. If there are multiple solutions, output any one of them.

The expression should contain between 1 and n fractions of the form a/b ($1 \leq a, b \leq n$). Fractions should be separated by "+" or "-" symbols. The "-" symbol may appear before the first fraction, but the "+" symbol may not.

The expression should not contain spaces or any other additional characters.

Example

standard input	standard output
2 3 6	1/2-1/3 -3/6+1/4+2/5-5/6+6/4-4/5

Note

In the first test case we have $n = 3$ and the expression `1/2-1/3` = `1/6`. It is impossible to obtain a positive number smaller than `1/6`. Other possible solutions include `-1/3+1/2`, or `2/3-1/2`, or `-3/2+1/1+2/3`.

In the second test case we have `-3/6+1/4+2/5-5/6+6/4-4/5` = `1/60` for $n = 6$.

Problem E. Express Rotations

Time limit:	3 seconds
Memory limit:	1024 megabytes

You are given a sequence of n numbers $a_1, a_2, \ldots, a_n$. Your task is to remove all elements in order from largest to smallest, and minimize the total cost of doing so. Equal elements can be removed in any order. You may perform three types of operations:

- Move the first element to the end of the sequence.
- Move the last element to the beginning of the sequence.
- Remove the first element, with the condition that there is no larger element in the sequence.

The cost of a move is equal to the value of the element being moved. Removal has no cost. Find the minimum possible total cost to obtain an empty sequence.

Input

The first line contains an integer n ($1 \le n \le 500\,000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$).

Output

Print a single integer – the minimum total cost.

Example

standard input	standard output
6 6 10 6 5 4 5	16

Note

The optimal sequence of operations is:

- $(6, 10, 6, 5, 4, 5)$, move the first element to the end. Cost 6.
- $(10, 6, 5, 4, 5, 6)$, remove the first element.
- $(6, 5, 4, 5, 6)$, remove the first element.
- $(5, 4, 5, 6)$, move the last element to the beginning. Cost 6.
- $(6, 5, 4, 5)$, remove the first element.
- $(5, 4, 5)$, remove the first element.
- $(4, 5)$, move the first element to the end. Cost 4.
- $(5, 4)$, remove the first element.
- (4), remove the first element.
- $()$, the sequence is empty.

The total cost is $6 + 6 + 4 = 16$.

Problem F. Frogs

Time limit:	2 seconds
Memory limit:	1024 megabytes

In a very long and narrow pond a mother frog has laid n eggs, the i-th egg at position a_i. From each egg a frog hatches (gets born) in the order from 1 to n. You are also given the jump length k.

At some moment the first p hatched frogs will start playing a game of frog-tag. In this game each frog indefinitely chases its younger sibling, and the youngest chases the oldest (frog i chases frog $i+1$, p chases 1). Every second each frog jumps either left or right by k in the direction of the frog it is chasing; it jumps right if they are on the same position. The frogs from $p+1$ to n do not participate in the game. Formally, for each of the p frogs, simultaneously: if $a_{1+(i \bmod p)} \geq a_i$ then a_i increases by k, otherwise a_i decreases by k.

The mother frog is concerned that her children might move too far away and jump out of the pond. Independently for each p from 2 to n, check if, in the game of frog-tag with frogs $1, 2, \ldots, p$, any one of them will ever move away from its initial position by at least $99^{(99^{99})}$. For each p output `1` if this happens, and `0` otherwise.

Input

The first line contains two integers n and k ($2 \leq n \leq 500\,000$; $1 \leq k \leq 10^9$) denoting the number of eggs and the jump length.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) denoting the positions of the eggs.

Output

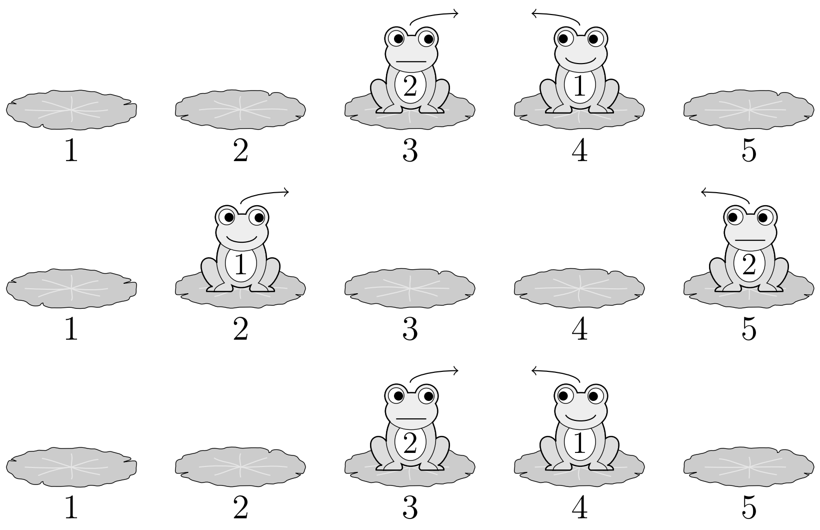
Output the answers for each $p = 2, 3, \ldots, n$ without spaces. If the first p frogs played frog-tag indefinitely, output `1` if any of them would move away from her initial position by at least $99^{(99^{99})}$, or output `0` otherwise. Thus the output should be a binary string of length $n-1$.

Example

standard input	standard output
6 2 4 3 -3 5 100 100	01011

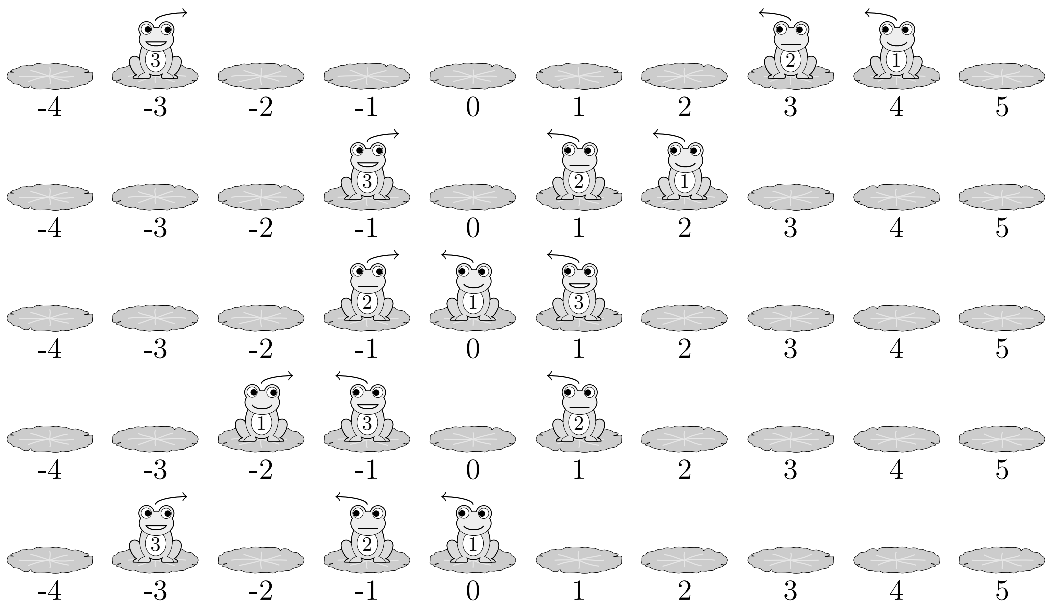
Note

The figures below (see also the second page) show the first few seconds of the frog-tag game for $p = 2, 3, 4$. For $p = 2$ two frogs start at positions 4 and 3, to which they return every 2 seconds. No frog moves very far from its starting position, so the answer is `0`.

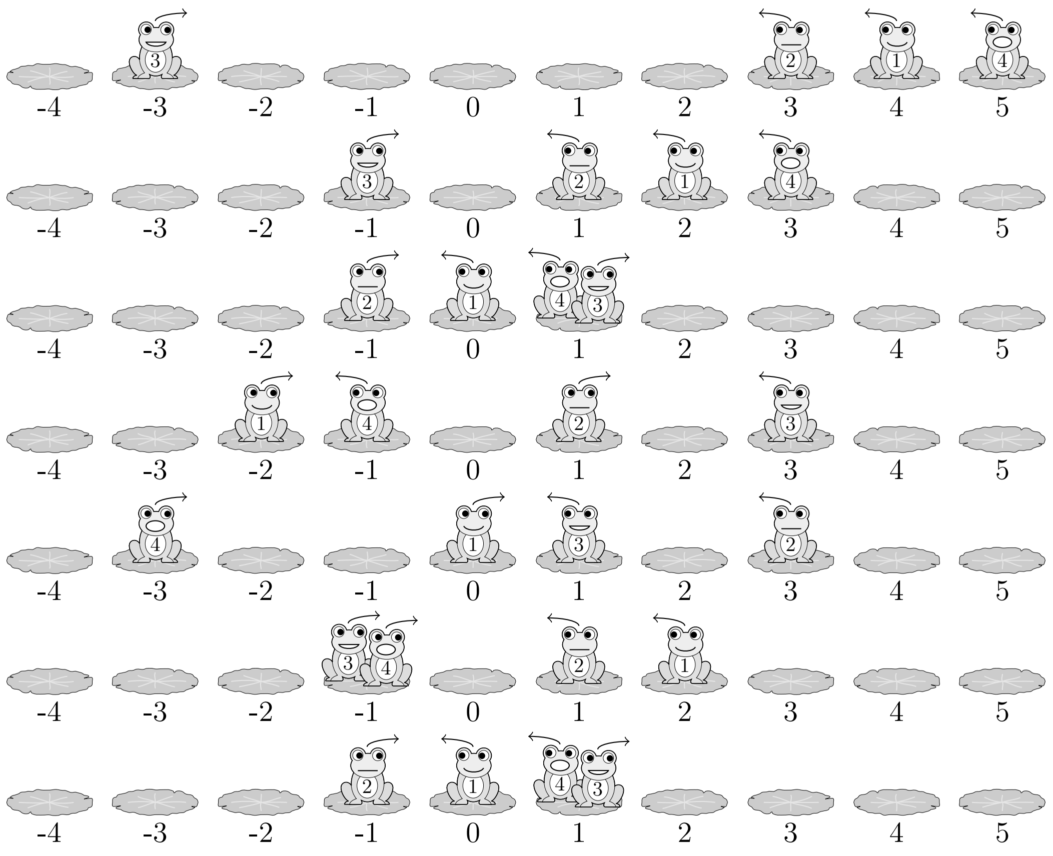


Figures for $p = 3$ and $p = 4$ are on the next page!

The first few seconds for $p = 3$, so $a = [4, 3, -3]$. The answer is `1`.



The first few seconds for $p = 4$, so $a = [4, 3, -3, 5]$. The answer is `0`.



Problem G. Game MPO

Time limit: 2 seconds
Memory limit: 1024 megabytes

In the board game MPO you earn points by building structures on the cells of an $n \times n$ square board. You are given the initial state of the board, where each cell is described by one of the 7 characters: "`.mpoMPO`":

- Uppercase letters `M`, `P`, `O` denote cells with built structures: metropolis, park, or ocean, respectively.
- Lowercase letters `m`, `p`, `o` denote empty cells where metropolis, park, or ocean, respectively, can be built.
- A dot '`.`' denotes an empty cell where no structure can be built.

You gain 1 point for each structure, and 1 point for each pair of adjacent (by side or corner) metropolis-park structures, or ocean-ocean structures. A *move* is building a structure on a permitted cell, which means changing a lowercase letter to its corresponding uppercase letter: `m` $\to$ `M`, `p` $\to$ `P`, or `o` $\to$ `O`. **It is forbidden to make a move that earns only one point** (that is, a move that increases the score by exactly 1).

Your strategy is simple. You always make one of the moves that earns most points. You stop playing when you cannot make a legal move – when no possible move earns at least 2 points.

Your task is to determine the initial and final score, as well as the final board state. It can be proven that the final state does not depend on how we resolve ties between moves that earn the same number of points.

Input

The first line contains an integer n ($2 \le n \le 10$) denoting the size of the board.

The next n lines describe the initial board. Each line contains a string consisting of n characters "`.mpoMPO`".

Output

In the first line output two integers – the initial score and the final score.

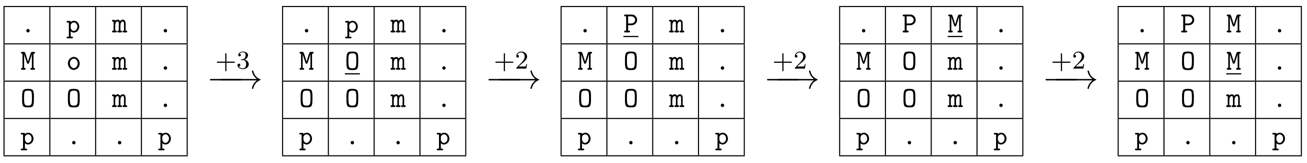
The following n lines should describe the final board in the same format as in the input.

Example

standard input	standard output
4 .pm. Mom. OOm. p..p	4 13 .PM. MOM. OOm. p..p

Note

The initial board has 3 structures (one metropolis and two oceans) and there is one pair of adjacent oceans, resulting in a total score of 4.



- First move: build an ocean, earning 3 points (1 for the new structure and 2 for the two new pairs of adjacent oceans).
- Second move: build a park, earning 2 points (1 for the new structure and 1 for a new metropolis-park pair).
- Third and fourth moves (in any order): build two new metropolises, each earning you 2 points.

At the end, there are three remaining cells where structures could be built, but a move on any of them would earn you only 1 point. The game ends with a total score of 13 points (7 structures, 3 metropolis-park pairs, 3 ocean-ocean pairs).

Problem H. High Jump

Time limit:	2 seconds
Memory limit:	1024 megabytes

Henry is participating in a high jump competition. Before each jump he can choose the height of the bar as an integer from 1 to n. As his long-time coach you know Henry's abilities well. For each possible height h you know the probability p_h of successfully clearing the bar at that height. Naturally, the greater the height, the lower the probability of success.

In today's competition there is no room for mistakes. One failed jump ends the competitor's performance, with his result being the highest previously cleared height (or 0 if he failed the first jump). The performance automatically ends with a score of n if height n is cleared. Help Henry choose the optimal heights for each jump. What is the maximum possible expected value[2] of his score?

Input

The first line contains an integer n ($1 \le n \le 500\,000$) denoting the limit on the bar height.

The second line contains n real numbers $p_1, p_2, \ldots, p_n$ ($0 < p_i < 1$; $p_i > p_{i+1}$), each with at most 9 digits after the decimal point. The number p_i is the probability of successfully clearing the bar at height i.

Output

Output a single real number – the maximum possible expected value of Henry's score.

Your answer should have an absolute or relative error of at most 10^{-6}. That means that if you output x, and the correct exact result is y, then your answer will be judged as correct if $|x - y| \le 10^{-6} \cdot \max(1, y)$. You may output up to 20 digits after the decimal point.

Example

standard input	standard output
5 0.9 0.85 0.6 0.456000 0.000000017	2.475200006589

Note

The following strategy is optimal:

- Set the bar at height 2. Henry clears it with probability 0.85 or ends with a score of 0 (probability 0.15).
- If he clears the first jump, set the bar at height 4. Henry clears it with probability 0.456 or ends with a score of 2.
- If he clears the second jump, set the bar at height 5. Henry either clears it with probability 0.000000017 and ends with a score of 5, or fails, ending with a score of 4.

The expected value of this strategy is:

$$0 \cdot 0.15 + 2 \cdot 0.85 \cdot 0.544 + 4 \cdot 0.85 \cdot 0.456 \cdot 0.999999983 + 5 \cdot 0.85 \cdot 0.456 \cdot 0.000000017 = 2.4752000065892$$

[2] *The expected value* is the probability-weighted average value of a random variable. Intuitively, it is the average outcome of a random experiment if it was repeated many times.

Problem I. Imbalanced Teams

Time limit:	4 seconds
Memory limit:	1024 megabytes

There are n players who regularly attend volleyball practice. Coach Igor knows their abilities well: a_i is the skill level of the i-th player. Each practice session consists of a series of matches, and for each match the coach selects two disjoint teams of k players each. The skill level of a team is defined as the sum of the skill levels of all its players.

The coach has noticed that the larger the difference (that is, the absolute value of the difference) between the teams' skill levels, the faster the match ends. Faster matches mean more games can be played in a single practice session! Thus he decided to select teams in such a way that maximizes the difference in their skill levels.

Help Igor plan the entire practice session. Find the maximum possible difference between the skill levels of two teams, and then count the number of matches that achieve this maximum, modulo $10^9 + 7$.

Matches cannot be repeated, meaning two teams can play against each other at most once. For example, for $n = 4$ and $k = 2$ there are only three possible matches: players 1 and 2 play against players 3 and 4; or players 1 and 3 play against players 2 and 4; or players 1 and 4 play against players 2 and 3.

Input

The first line contains two integers n and k ($2 \le n \le 2000$; $1 \le k \le \frac{n}{2}$) representing the number of available players and the size of each team respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$), the skill levels of each player.

Output

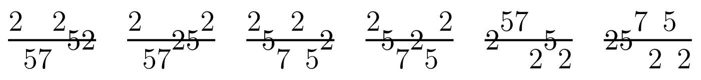
Output two integers – the maximum difference in team skill levels and the remainder when the number of matches that achieve this maximum is divided by $10^9 + 7$.

Examples

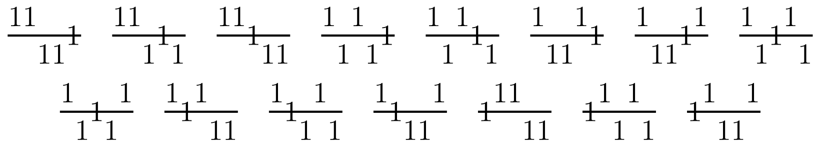
standard input	standard output
6 2 2 5 7 2 5 2	8 6
5 2 1 1 1 1 1	0 15

Note

In the first sample test the coach selects two teams of $k = 2$ players for each match. He can organize 6 matches with a skill level difference of 8. The matches are shown in the diagram below. For example, in the first match players 1 and 4 play against players 2 and 3, and the skill level difference is: $|(a_1 + a_4) - (a_2 + a_3)| = |(2 + 2) - (5 + 7)| = 8$.



In the second sample test all players have the same skill level, so the difference between the teams' skill levels is always 0. Thus all 15 possible matches have the maximum difference of 0:



Problem J. Just Zeros

Time limit:	1.5 seconds
Memory limit:	1024 megabytes

Julia has a rectangular grid consisting of $h \cdot w$ bits arranged in h rows and w columns, where the number of rows is small ($h \leq 8$). Rows are numbered from 1 to h from top to bottom, and columns from 1 to w from left to right. A bit is a value 0 or 1, and *negating* a bit means flipping its value to the opposite: 0 to 1 or 1 to 0.

The grid allows the following three types of operations:

- `P` i j – negate the bit at the intersection of the i-th row and j-th column,
- `R` i – negate all bits in the i-th row,
- `K` j – negate all bits in the j-th column.

Julia wants to clear the grid, i.e., turn all the bits into 0. The minimum number of operations required to achieve this is called the *difficulty* of the grid.

Mischievous Romek likes to disrupt Julia's plans and performs a total of q operations of one of the three types above. However, Romek doesn't realize that Julia enjoys such challenges. She observes the grid and calculates its difficulty at each of the $q+1$ moments (initially and after each of Romek's operations). Can you also determine these values?

Romek modifies the grid permanently. Julia does not perform any operations herself and does not change the grid.

Input

The first line contains three integers h, w and q ($1 \leq h \leq 8$; $1 \leq w, q \leq 10^5$) denoting the dimensions of the grid and the number of operations performed by Romek.

The next h lines describe the initial grid: each line contains a binary string (characters `0` and `1`) of length w.

The final q lines describe Romek's operations, each in the format: "`P` i j", or "`R` i", or "`K` j" ($1 \leq i \leq h$; $1 \leq j \leq w$).

Output

Output $q+1$ integers (one per line) – the difficulty of the initial grid followed by the difficulty of the grid after each of Romek's q operations.

Example

standard input	standard output
3 4 6 1010 1101 0010 R 2 P 3 1 K 2 P 2 1 K 4 P 3 4	3 2 3 4 3 3 4

Note

The figure illustrates the initial grid and the first few operations by Romek. The initial difficulty of the grid is 3 because Julia can use the following operations: `P 1 1`, `R 2`, `K 3`.

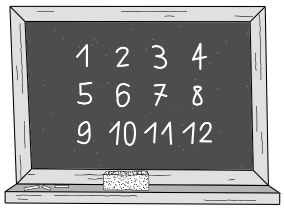
```
            1010            1010             1010            1110            1110
            1101    R 2 ->  0010   P 3 1 ->  0010    K 2 ->  0110   P 2 1 -> 1110    K 4 ->    ...
            0010            0010             1010            1110            1110

difficulty:   3               2                3               4               3
```

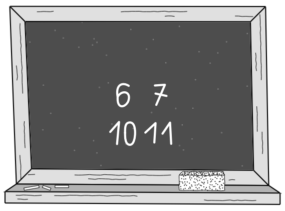
Problem K. Kindergarten Square

Time limit:	3 seconds
Memory limit:	1024 megabytes

Children in kindergarten are learning about natural numbers. The teacher wrote on the board consecutive numbers from 1 to $h \cdot w$ arranged in h rows and w columns ($h, w \geq 2$). The first row thus contains numbers from 1 to w (from left to right), the second row contains numbers from $w + 1$ to $2 \cdot w$, and so on. After the lesson the teacher erased all the numbers except for a 2×2 contiguous square containing four adjacent numbers from the original layout.

The drawing shows the board for $h = 3$, $w = 4$, and an example of a remaining 2×2 square:





One of the kindergartners, Kamilek, didn't pay much attention during the lesson and is now wondering which numbers h and w the teacher used. Find any possible pair (h, w) or output the number `-1` if the given square is impossible to obtain. The situation repeats over t days, given as independent test cases.

Input

The first line contains an integer t ($1 \leq t \leq 10$) denoting the number of test cases.

The next $2 \cdot t$ lines describe the test cases. Each test case consists of two rows, each with two integers in the range $[1, 50\,000]$ representing the remaining 2×2 square.

Output

Output t lines, the i-th with the solution for the i-th test case:

- If the given square is possible to obtain output two integers separated by a space – any possible dimensions of the board h, w ($2 \leq h, w \leq 100\,000$). It can be proven that if there exists any valid pair (h, w), there also exists a pair that meets these inequalities. If there are multiple solutions output any one of them.
- If the square could not have been obtained output the single number `-1`.

Example

standard input	standard output

4
6 7
10 11
2 3
4 5
8 5
5 13
1 2
5 6
	3 4
	-1
	-1
	3 4

Note

In the first test case we have the square $[[6, 7], [10, 11]]$. The drawing above shows one of the many correct solutions: $h = 3, w = 4$. This is also one of the correct solutions for the fourth test case $[[1, 2], [5, 6]]$.

In the second test case the numbers $[[2,3],[4,5]]$ might initially be written by the teacher, but they never form a 2×2 square. Thus the answer is `-1`.

Problem L. Looping RPS

Time limit:	4 seconds
Memory limit:	1024 megabytes

In this task: `K` is rock, `P` is paper, and `N` is scissors (from Polish words *kamie* , *papier*, *noyce*).

In a rock-paper-scissors duel two players simultaneously show one of the three signs. When two players show different signs, the duel ends with a victory according to the rule: paper beats rock, rock beats scissors, scissors beats paper. If both players show the same sign, they try again. A duel may last indefinitely, resulting in a draw.

Today is the day of AMPPKN[3]. At such a high level no one draws conclusions from the opponent's moves, afraid of cunning strategies that might exploit that. Randomizing moves is also difficult, so instead each competitor has written a strategy on their arm before starting – a word s consisting of the letters `K`, `P`, and `N`. In each duel a competitor repeats their strategy from the first sign in an infinite loop, for example `KPP` means playing in sequence: rock, paper, paper, rock, paper, paper, and so on.

There are n competitors in AMPPKN, with the i-th competitor using a strategy described by the word s_i. The organizers are interested in finding triples of competitors that behave like rock-paper-scissors, meaning each competitor would win against one of the other two.

Formally, count unordered triples of competitors such that these three competitors, in some order (A, B, C), satisfy that A would win against B, B would win against C, and C would win against A.

Input

The first line contains an integer n ($3 \le n \le 10^5$).

Each of the next n lines contains a non-empty word s_i consisting of the letters `P`, `K`, `N`. The total length of all words does not exceed 10^6.

Output

Output a single integer – the number of such triples of competitors.

Example

standard input	standard output
6 P PN KK N PKK PN	6

Note

There are 6 such triples:

`(P, KK, N), (P, PKK, PN), (P, PKK, PN'), (PN, KK, N), (KK, N, PKK), (KK, N, PN'),`

where `PN'` denotes the second occurrence of the word `PN`.

For example, in the second triple: `P` would win against `PKK` (repeated paper beats rock in the second move), `PKK` would win against `PN`, and `PN` would win against `P`.

[3]Akademickie Mistrzostwa Polski w Papier-Kamie -Noyce