The 4th Universal Cup

Stage 8: Grand Prix of Poland

December 6-7, 2025

This problem set should contain 13 problems on 21 numbered pages.

Based on

Akademickie Mistrzostwa Polski w Programowaniu Zespoowym (AMPPZ)

Hosted by

Problem A. AIMPPZ

Time limit:	1 second
Memory limit:	1024 megabytes

Alice has climbed the corporate ladder for many years and finally reached the very top. She has just become the Chief Executive Officer (CEO) of a large publicly traded company. The previous CEO was fired due to poor financial results, but Alice knows perfectly well how to boost profits. Time to choose a new company name!

Investors love any mention of artificial intelligence, especially in the company name. The base value of the company equals the length of its name, and each occurrence of the substring “`AI`” multiplies the company’s value by 5. For example, the name “$\texttt{A}\underline{\texttt{AI}}\texttt{CPC}\underline{\texttt{AI}}\texttt{IA}$” has length 10 and two (underlined) occurrences of “`AI`”:

- value(“`AAICPCAIIA`”) $= 10 \cdot 5 \cdot 5 = 250$

More examples:

- value(“`AXI`”) $= 3$ (zero occurrences of “`AI`”)
- value(“`AIAIAIAI`”) $= 8 \cdot 5 \cdot 5 \cdot 5 \cdot 5 = 5000$ (four occurrences of “`AI`”)
- value(“`AIMPPZ`”) $= 6 \cdot 5 = 30$ (one occurrence of “`AI`”)

Alice’s contract guarantees her an exceptionally generous bonus if the company’s value is **exactly** n. She also wants the name to be as short as possible.

Alice is already busy with her next task – designing the refreshed logo. Can you propose a new company name in the meantime? Find the shortest possible word of value n, consisting of uppercase English letters `A-Z`.

Input

The only line contains an integer n ($1 \leq n \leq 100\,000$) – the desired company value.

Output

Print a word consisting of the letters `A-Z` – a name with value n and the smallest possible length. If many solutions exist, output any of them.

Examples

standard input	standard output
30	AIMPPZ
250	AAICPCAIIA

Note

In the first test, we look for the shortest possible name of value 30. Optimal words have length 6 and one occurrence of “`AI`”, for example “`AIMPPZ`”, “`HUAWAI`”, “`BRAINS`” and “`AAAIII`”. No shorter word has value 30.

In the second test, value 250 cannot be achieved with a name shorter than 10 characters.

Problem B. Beats

Time limit:	1 second
Memory limit:	1024 megabytes

Bartek is a well-known techno music producer, and he is currently perfecting his next hit. He has created a track consisting of n beats, numbered consecutively from 1 to n. After spending hours in a music editing program, he was about to export the mp3 file, but he noticed that the beats are currently arranged in the order $a_1, a_2, \ldots, a_n$. Bartek needs to obtain an increasing order of beats, that is, $1, 2, \ldots, n$.

The program allows two types of operations:

- Move the last beat to any position, e.g. $(2, 4, 1, 5, \underline{3}) \to (2, \underline{3}, 4, 1, 5)$.
- Move any beat to the beginning, e.g. $(2, 4, 1, \underline{5}, 3) \to (\underline{5}, 2, 4, 1, 3)$.

What is the minimum number of operations needed to arrange the beats in increasing order?

Input

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

The second line contains n pairwise distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$; $a_i \ne a_j$), representing the current order of beats in the program.

Output

Output a single integer – the minimum number of operations required.

Examples

standard input	standard output
6 3 2 5 4 1 6	4
9 3 1 2 5 6 4 9 7 8	6

Note

We start with the sequence of beats $(3, 2, 5, 4, 1, 6)$. Here is one way to achieve the desired order in four operations:

- Move the last beat two places to the left: $(3, 2, 5, \underline{6}, 4, 1)$.
- Move beat 2 to the beginning: $(\underline{2}, 3, 5, 6, 4, 1)$.
- Move beat 1 to the beginning: $(\underline{1}, 2, 3, 5, 6, 4)$.
- Move the last beat two places to the left: $(1, 2, 3, \underline{4}, 5, 6)$.

It is possible to obtain an increasing order in other ways using four operations, but it cannot be done in three operations, so the result is 4.

Problem C. Count Triangular Sequences

Time limit:	1 second
Memory limit:	1024 megabytes

A number n is given. Your task is to count all *triangular sequences* and output the remainder of dividing their count by $10^9 + 7$. A sequence is called triangular if it satisfies the following conditions:

- The length of the sequence does not exceed n.
- Each element is an integer from the interval $[1, n]$.
- Some three elements of this sequence can be the lengths of the sides of a non-degenerate triangle[1].

Example triangular sequences for $n = 100$ are $(1, 1, 1)$ and $(16, 1, 11, 25, 100)$. In the first, the sides of the non-degenerate triangle are $(1, 1, 1)$, and in the second, they are $(16, 11, 25)$.

The sequence $(16, 1, 11, 84, 100)$ is not triangular, because no three elements form the sides of a non-degenerate triangle. The sequence $(5, 5)$ is also not triangular.

Input

The only line contains an integer n ($3 \le n \le 200\,000$).

Output

Print a single integer – the remainder of dividing the number of triangular sequences by $10^9 + 7$ (that is, 1000000007).

Examples

standard input	standard output
3	15
4	254

Note

For $n = 3$ there are 15 triangular sequences: (1,1,1), (1,2,2), (1,3,3), (2,1,2), (2,2,1), (2,2,2), (2,2,3), (2,3,2), (2,3,3), (3,1,3), (3,2,2), (3,2,3), (3,3,1), (3,3,2), (3,3,3).

For $n = 4$ there are 254 triangular sequences: (1,1,1), (1,1,1,1), (1,1,1,2), (1,1,1,3), (1,1,1,4), (1,1,2,1), …, (4,4,2,4), (4,4,3), (4,4,3,1), (4,4,3,2), (4,4,3,3), (4,4,3,4), (4,4,4), (4,4,4,1), (4,4,4,2), (4,4,4,3), (4,4,4,4).

[1]A triangle is non-degenerate if it has a positive area. Equivalently, its vertices are not collinear.

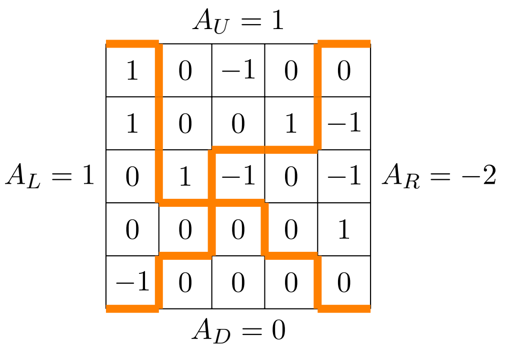
Problem D. Division with Polylines

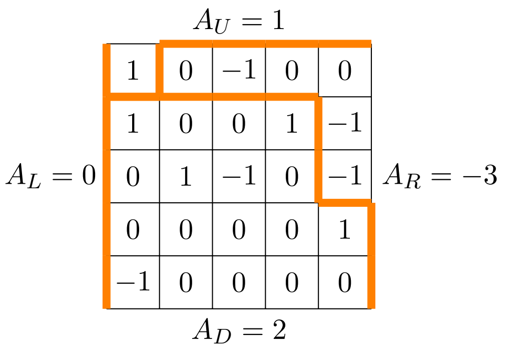
Time limit:	4 seconds
Memory limit:	1024 megabytes

You are given a board of size $n \times n$, with each of its n^2 cells containing one of the numbers -1, 0, or 1. You are also given four integers A_L, A_D, A_R and A_U. We want to draw two polylines:

- one connecting the top-left corner of the board with the bottom-right corner; moving along the edges of the cells only to the right and down,
- and the other connecting the bottom-left corner of the board with the top-right corner, moving only to the right and up.

The two polylines divide the board into four (possibly empty) regions: left, down, right, and up. Check whether there exist two polylines, such that the cells in these four regions sum up to A_L, A_D, A_R and A_U, respectively. Print `TAK` (yes) or `NIE` (no) for each of the t independent test cases.





The figures show the same 5×5 board from the sample test.

The left figure shows a partition into regions with sums $(1, 0, -2, 1)$ (in the order A_L, A_D, A_R, A_U). The first polyline can be described by the word `RDDDRRDRDR`, and the second by `RURUURRUUR` (where `D`, `U`, and `R` denote moves along edges down, up, and right, respectively). For example, the right region contains 8 cells with a sum $A_R = 0 - 1 - 1 + 0 - 1 + 0 + 1 + 0 = -2$.

The right figure shows a partition into regions with sums $(0, 2, -3, 1)$. The left region is empty, so the sum of its cells is zero. The polylines can be described by the words `DRRRRDDRDD` and `UUUURURRRR`.

The same board cannot be partitioned into regions with sums $(-2, 0, 1, 1)$, i.e. $A_L = -2$, $A_D = 0$, $A_R = 1$, $A_U = 1$. Note that the same values, but in a different order, appear in the left figure – the order matters.

Input

The first line of input contains a single integer t ($1 \le t \le 100\,000$), denoting the number of test cases.

The first line of a test case contains five integers n, A_L, A_D, A_R, A_U ($1 \le n \le 2000$; $-n^2 \le A_L, A_D, A_R, A_U \le n^2$), denoting the side length of the board and the desired sums of the four regions.

Each of the next n lines of the test case describes the board; the i-th line contains n integers $a_{i1}, a_{i2}, \ldots, a_{in}$ ($a_{ij} \in \{-1, 0, 1\}$), describing the i-th row. The total sum of the board is equal to $A_L + A_D + A_R + A_U$.

The total sum of n^2 over all test cases does not exceed $4\,000\,000$.

Output

Output exactly t lines – for each test case output the word `TAK` (*yes*) if there exist polylines dividing the board into regions with the specified sums, or the word `NIE` (*no*) otherwise.

Example

standard input	standard output
`5`	`TAK`
`5 1 0 -2 1`	`TAK`
`1 0 -1 0 0`	`NIE`
`1 0 0 1 -1`	`NIE`
`0 1 -1 0 -1`	`TAK`
`0 0 0 0 1`	
`-1 0 0 0 0`	
`5 0 2 -3 1`	
`1 0 -1 0 0`	
`1 0 0 1 -1`	
`0 1 -1 0 -1`	
`0 0 0 0 1`	
`-1 0 0 0 0`	
`5 -2 0 1 1`	
`1 0 -1 0 0`	
`1 0 0 1 -1`	
`0 1 -1 0 -1`	
`0 0 0 0 1`	
`-1 0 0 0 0`	
`2 1 -1 0 1`	
`1 0`	
`-1 1`	
`1 0 0 0 0`	
`0`	

Note

The first three test cases correspond to the examples shown in the statement. They contain the same board and queries for sums (A_L, A_D, A_R, A_U) in order:

1. $(1, 0, -2, 1)$, answer `TAK`
2. $(0, 2, -3, 1)$, answer `TAK`
3. $(-2, 0, 1, 1)$, answer `NIE`

Problem E. Enigma

Time limit:	2 seconds
Memory limit:	1024 megabytes

Edward is the founder and head of the company Enigma, the manufacturer of the most secure safes on the market. Enigma safes are protected by a combination lock with a 5-digit code. The lock consists of five wheels and an "open" button. Each wheel contains all digits from 0 to 9 (in this order); after the digit 9, the digit 0 appears again.

An *operation* is defined as rotating any wheel by one position in either direction, for example changing the digit 9 to 8 or 0. The *distance* between two codes is the minimum possible number of operations required to change one entered code into the other. For example, the distance between `38911` and `68000` is $3 + 0 + 1 + 1 + 1 = 6$.

Safe manufacturers keep statistics of the most commonly used codes – you are given n such codes in order from most to least popular. Enigma prides itself on flexibility – each client can choose how many of the n most popular codes they wish to avoid. Edward then chooses a code that maximizes the smallest distance to the codes from the selected prefix of the sequence of the most popular codes.

For each k from 1 to n, your task is to find the largest possible distance of some code to the closest among the k most popular codes, and count how many codes achieve such a distance.

Input

The first line contains an integer n ($1 \le n \le 99\,999$), denoting the number of popular codes.

Each of the following n lines contains a string consisting of 5 digits `0-9`, possibly with leading zeros. These are the popular codes in order from most to least popular. All codes are pairwise distinct.

Output

Output n lines; the k-th line should contain two integers separated by a space – the largest possible distance to the nearest code among the k most popular ones, and the number of codes with that distance.

Examples

standard input	standard output

3
00000
00001
12345
	25 1
	24 2
	17 270

3
12345
23456
67890
	25 1
	22 20
	12 15270

Note

For $k = 1$, the largest possible distance from code `00000` is 25, achieved only by the code `55555`.

For $k = 2$, the largest possible distance to the nearer of the codes `00000` and `00001` is 24, achieved by two codes: `55555` and `55556`. The first is 24 away from `00001`, and the second is 24 away from `00000`.

For $k = 3$, the largest possible distance to the nearest among the codes `00000`, `00001`, and `12345` is 17, achieved by 270 different codes, for example `55570` and `55580`. The first is 17 away from `12345`, and the second is 17 away from `00000`.

Problem F. Finances

Time limit: 5 seconds
Memory limit: 1024 megabytes

Tomorrow (November 17th) we celebrate Debt-Free Day. On this occasion, n friends decided to finally settle their mutual debts from many past trips. They always recorded their expenses in an app, which now summarized and simplified everything. For each person, it displays a total balance a_i – positive if that person is owed money, or negative if that person must repay money. The sum of all displayed values is, of course, equal to zero: $\sum_{i=1}^{n} a_i = 0$. We want to plan transfers (with integer values) in such a way that each person's balance becomes zero.

The n people are connected by m trust relationships, described by triplets (u_j, v_j, c_j) – persons u_j and v_j trust each other enough that either of them can send the other one transfer of value at most c_j. People not directly connected by a trust relationship cannot send money to each other. Unordered pairs $\{u_j, v_j\}$ do not repeat, and the entire trust network is connected – every pair of people is linked by a (possibly multi-step) path of trust relationships.

The app displays the total debt A (equivalently, the sum of positive a_i values). In this group of friends, the trust levels are quite high – for every pair connected by a trust relationship, it holds that $c_j \geq \lceil \frac{A}{2} \rceil$. For example, for the sequence $a = (10, -5, 0, 3, -4, -4)$ we have $A = 5 + 4 + 4 = 10 + 3 = 13$, hence $c_j \geq 7$.

Can this group of n people agree on directions and amounts of transfers to fully settle all debts – that is, make everyone's balance zero? Print `TAK` (yes) or `NIE` (no) for each of the t independent test cases.

You may assume that every person has enough money to perform any necessary transfers.

Input

The first line contains an integer t ($1 \leq t \leq 10^5$), denoting the number of test cases.

The first line of each test case contains two integers n and m ($2 \leq n \leq 10^6$; $n-1 \leq m \leq 10^6$), representing respectively the number of people and the number of trust relationships.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$; $\sum_{i=1}^{n} a_i = 0$), representing the initial balance of each person. At least one of the numbers a_i is nonzero.

We define $A = \sum_{i=1}^{n} \max(0, a_i)$. From the previous condition it follows that $A \geq 1$.

The next m lines describe the trust relationships; the j-th line contains three integers u_j, v_j, and c_j ($1 \leq u_j, v_j \leq n$; $u_j \neq v_j$; $\lceil \frac{A}{2} \rceil \leq c_j \leq 10^{15}$). The network of n people and m trust relationships is connected, and each unordered pair $\{u, v\}$ appears in a given test case at most once.

The sum of all n values across test cases does not exceed 10^6; likewise, the sum of all m values does not exceed 10^6.

Output

Print t lines – for each test case, output the word `TAK` if it is possible to balance all accounts, or the word `NIE` otherwise.

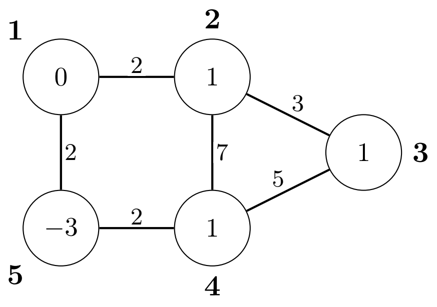
Example

standard input	standard output
3	NIE
2 1	TAK
20 -20	TAK
1 2 10	
5 6	
0 1 1 1 -3	
1 2 2	
2 3 3	
3 4 5	
2 4 7	
1 5 2	
4 5 2	
3 2	
1 2 -3	
1 2 1000000000000000	
2 3 999997235527681	

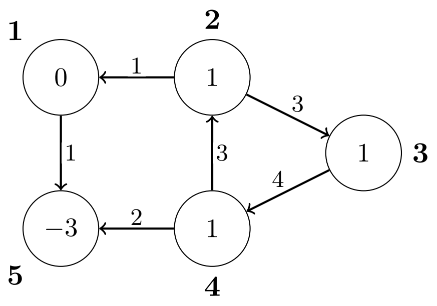
Note

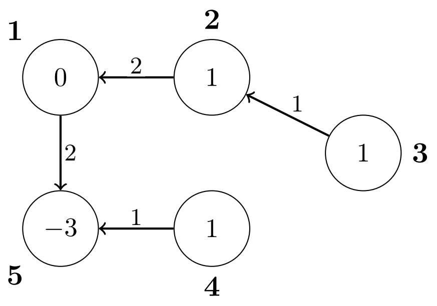
In the first test case, there are two people with balances $a = (20, -20)$. The first person should send the second one a transfer of value 20, but can send at most $c_1 = 10$. The answer is NIE (no).

The figure below illustrates the second test case with balances $a = (0, 1, 1, 1, -3)$ written inside the vertices, and limits c_j on the edges.



Below are two example valid transfer plans. In both plans, transfer limits are not exceeded, and everyone ends up with balance 0. The answer is TAK (yes).





Problem G. Game of Pipes

Time limit:	3 seconds
Memory limit:	1024 megabytes

This is an interactive problem.

The game "Plumber Mariusz" consists of $n+1$ levels (numbered from 0 to n), each of which has k versions (numbered from 0 to $k-1$). Each version of every level (except the last one) ends with two pipes – the left and the right – and Mariusz must jump into one of them. Each pipe contains some number of coins and leads to one of the versions of the next level. **Exactly two pipes lead to each version of each level (except level zero)**; these may be any two among the $2 \cdot k$ pipes coming out of all k versions of the previous level.

A single playthrough consists of choosing a version of level zero and then choosing left or right pipe n times, and the *result* of the playthrough is the total number of coins collected in the n visited pipes. We do not learn how many coins were in each pipe or which versions of the levels were visited. Coins in the pipes always reset to their initial values, so all playthroughs are independent. A playthrough can therefore be represented by the function:

`int rozgrywka(int ver, string s)`, where $ver \in [0, k-1]$ and s is a string of n letters `L` and `P`.

Note that `L` and `P` correspond to the polish words for *left* and *right* respectively.

You may perform up to 300 000 test playthroughs and learn their results. After that, you will receive up to 10 000 queries about playthroughs. Your task is to determine their results.

Each test is fixed before the interaction begins (i.e. the interactor is not adaptive). A test contains a directed weighted graph with $(n+1) \cdot k$ vertices arranged in $n+1$ layers (levels), and q queries (ver, s). Each vertex (except those in the last layer) has two outgoing edges (left and right), each described by a coin count $c \in [1, 10^8]$ and a version $v \in [0, k-1]$ it leads to. Recall that every vertex except those in layer zero has in-degree exactly 2.

Buffer flushing

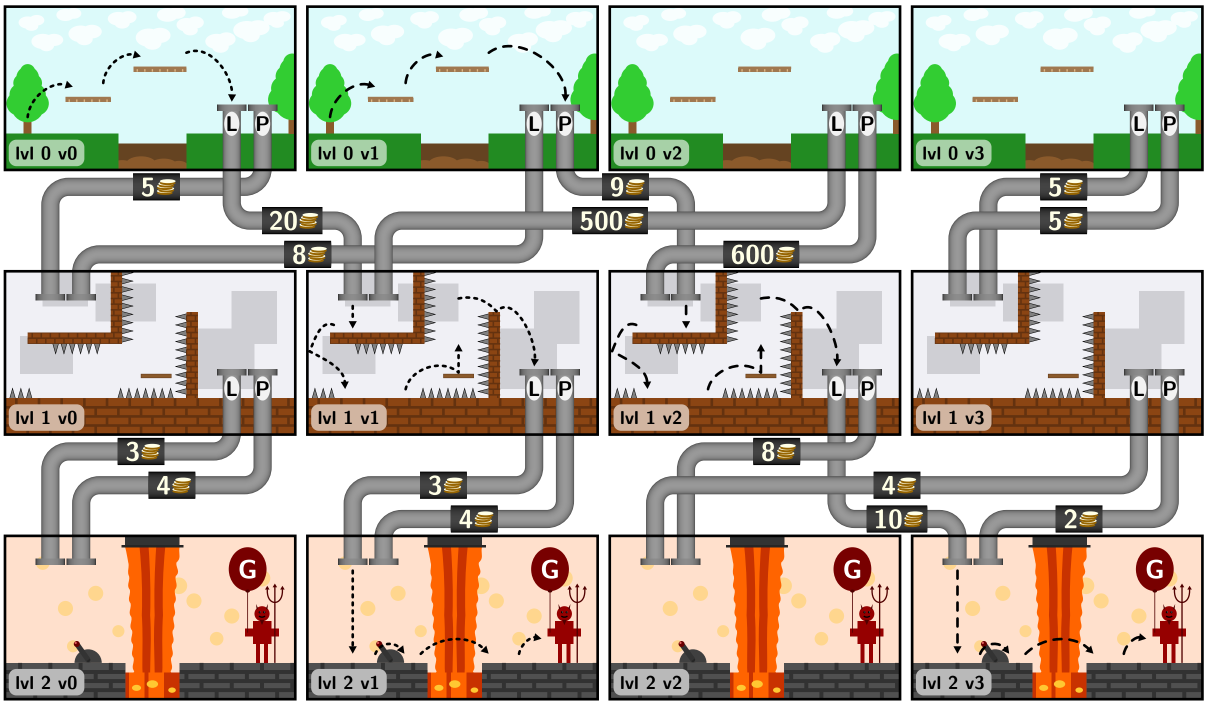
After outputting each line you must flush the output buffer, so that the interactor receives the output line. For example, in C++ you may output a line containing an exclamation mark using `cout << "!" << endl;` or using `printf("!\n"); fflush(stdout);`. In Python you may use `print("!", flush=True)`. We do not guarantee stable behavior of the interactor if you output multiple lines without flushing after each of them, or if you output many lines without reading the interactor's responses. Your program may not open any files.

Interaction Protocol

The full communication flow is described below:

- Read the first line containing two integers n and k ($1 \le n, k \le 20$) – there are $n+1$ levels, each with k versions.
- Up to 300 000 times, perform a test playthrough:
  - Output a line of the form „`? ver s`" (without quotes), where $ver \in [0, k-1]$ and s is a string of n letters `L` and `P`. **Flush**.
  - Read the next line containing the result of the playthrough – an integer x ($n \le x \le n \cdot 10^8 \le 2 \cdot 10^9$).
- Output a line containing a single character `!` (exclamation mark). **Flush**.
- Read the next line containing an integer q ($1 \le q \le 10\,000$) – the number of queries.

- For each of the q queries:
  - Read the next line containing an integer ver and a string s (same limits as above).
  - Output a line containing the result of the playthrough (an integer). **Flush**.



The illustration shows the first sample test ($n = 2$; $k = 4$). Two final queries are marked using dashed lines: `rozgrywka(0, "LL")` $= 20 + 3 = 23$ and `rozgrywka(1, "PL")` $= 9 + 10 = 19$.

Example

Here is an example communication flow:

Your program	Interactor	Description
	`2 4`	3 levels $(0 \ldots 2)$, each with 4 versions $(0 \ldots 3)$.
`? 0 LL`		
	`23`	First dashed path in the figure. Start from version 0 of level 0, jump into the left pipe (20 coins). From version 1 of level 1 again jump into the left pipe (3 coins). Result: $20+3 = 23$.
`? 0 LL`		
	`23`	Same playthrough, same result.
`? 2 PL`		
	`610`	$600 + 10 = 610$.
`? 0 PP`		
	`9`	$5 + 4 = 9$.
`? 3 LP`		
	`7`	$5 + 2 = 7$.
`!`		End of test playthroughs.
	`2`	There are $q = 2$ queries to answer.
	`0 LL`	Lucky – we tested this exact playthrough!
`23`		
	`1 PL`	We do not know the result and cannot deduce it. We never tested starting from version 1 of level 0.
`19`		By chance, we guess the correct result: $9 + 10 = 19$.

Note

In the attachment (you can download it in the "attachment" tab at QOJ) you will find sample (incorrect) solutions `G.cpp` and `G.py` that perform the communication correctly, but compute the results incorrectly. The directory also includes: a sample checker `Gsoc.cpp` and sample tests (`G0a.in`, `G0b.in`, `G0c.in`, and a large random test `G0d.in` with $n = k = 20$). The format of test files is described in the comment at the beginning of `Gsoc.cpp`.

The sample checker `Gsoc.cpp` differs from the one used in QOJ. It might not validate input or the arguments of the playthrough function.

You can run your solution using the script `run.sh`. The command takes the compiled program and the test file name. For example, to run the sample programs on the sample test `G0a.in`:

- C++: `g++ G.cpp -o G.e && ./run.sh "./G.e" G0a.in`
- Python: `./run.sh "python3 G.py" G0a.in`

Problem H. Hacking

Time limit:	1 second
Memory limit:	1024 megabytes

Yesterday at 9:00, the mandatory activities related to the AMPPZ contest began. The schedule contained n activities that followed one after another, lasting $t_1, t_2, \ldots, t_n$ minutes respectively. Yesterday at 19:00, the famous hacking contest also took place, and by that time, the participants wanted to already be in the hotel.

Determine how much time the participants had to return to the hotel after the last activity, assuming everything went according to plan yesterday.

Input

The first line contains an integer n ($1 \le n \le 100$), denoting the number of activities.

The second line contains a sequence of n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le 500$), denoting the durations of the consecutive activities in minutes.

You may assume that the organizers ensured that all activities in the schedule ended no later than yesterday at 18:59, leaving the participants a positive amount of time to return to the hotel.

Output

Print a single integer – the number of minutes the participants had left to return to the hotel.

Example

standard input	standard output
10 45 15 180 60 30 15 105 15 30 45	60

Note

The consecutive activities in the schedule were as follows:

Hours	**Activity**
09:00–09:45	Opening
09:45–10:00	Break
10:00–13:00	Lecture session
13:00–14:00	Lunch
14:00–14:30	Contest organization rules
14:30–14:45	Break
14:45–16:30	Practice contest
16:30–16:45	Break
16:45–17:15	Practice contest discussion
17:15–18:00	Dinner

Exactly 60 minutes remained to return to the hotel before 19:00.

Problem I. ICPC Isolation

Time limit:	2 seconds
Memory limit:	1024 megabytes

In AMPPZ 2025, 80 teams from 27 institutions are competing. The abbreviations of these institutions are UW, UJ, UWR, MAP, PW, AGH, PG, NLU, PUT, PO, PWR, SGGW, UMCS, UR, ZUT, DTP, GOO, HUA, KUL, PL, PM, PS, UAM, UG, UMK, UO, WAT. Here is the complete table showing the number of teams from each institution[2]:

full name	abbreviation	number of teams
Uniwersytet Warszawski	UW	11
Uniwersytet Jagiello ski	UJ	9
Uniwersytet Wrocawski	UWR	8
Mistrzostwa w Algorytmice i Programowaniu (secondary schools)	MAP	7
Politechnika Warszawska	PW	6
Akademia G ó rniczo-Hutnicza im. Stanisawa Staszica w Krakowie	AGH	5
Politechnika Gda ska	PG	4
Wysza Szkoa Biznesu - National-Louis University	NLU	3
Politechnika Pozna ska	PUT	3
Politechnika Opolska	PO	2
Politechnika Wrocawska	PWR	2
Szkoa G ó wna Gospodarstwa Wiejskiego w Warszawie	SGGW	2
Uniwersytet Marii Curie-Skodowskiej w Lublinie	UMCS	2
Uniwersytet Rzeszowski	UR	2
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie	ZUT	2
Digital Technology Poland	DTP	1
Google	GOO	1
Huawei	HUA	1
Katolicki Uniwersytet Lubelski Jana Pawa II	KUL	1
Politechnika ó dzka	PL	1
Politechnika Morska w Szczecinie	PM	1
Politechnika lska	PS	1
Uniwersytet im. Adama Mickiewicza w Poznaniu	UAM	1
Uniwersytet Gda ski	UG	1
Uniwersytet Mikoaja Kopernika w Toruniu	UMK	1
Uniwersytet Opolski	UO	1
Wojskowa Akademia Techniczna	WAT	1

The contest hall is an 8×10 grid consisting of 8 rows with 10 workstations each. To prevent unauthorized communication, the organizers must find an arrangement in which teams from the same institution do not sit next to each other – neither sharing an edge nor a corner.

The audience has its favorite – a team from the institution whose abbreviation is given in the input. The film crew has already set up their cameras in the top-left corner of the contest hall, so one team from the institution given in the input must be seated there.

Find any arrangement of teams that satisfies the given conditions.

Input

The only line contains a word consisting of 2 to 4 uppercase letters – the abbreviation of the favorite institution (one of the 27 listed above).

[2]The list of teams may change slightly between the preparation of this task and the contest.

Output

Print the found arrangement of teams – 8 lines, each with 10 words (abbreviations) separated by spaces. The first word in the first line must match the favorite institution provided in the input. If multiple solutions exist, print any of them.

Note

In the example test, the audience's favorite is Uniwersytet Wrocawski[3]. The provided output contains correct abbreviations and counts, and UWR indeed appears in the top-left corner. However, **this is not a valid output**, because in many places teams from the same institution are adjacent to each other. Below, three example collisions are highlighted in gray.

UWR	MAP	MAP	UJ	PL	UW	UW	SGGW	ZUT	GOO
MAP	PUT	UR	KUL	UW	UW	PM	AGH	UWR	AGH
PS	UJ	UWR	UW	UWR	MAP	UWR	PG	UWR	PWR
PW	HUA	PW	AGH	PG	DTP	UW	UW	UW	PWR
MAP	UG	NLU	UJ	PUT	UAM	UW	UJ	UJ	UW
NLU	ZUT	PW	WAT	UWR	AGH	UMCS	UR	UJ	AGH
PUT	UJ	SGGW	PG	PW	MAP	PO	MAP	UJ	UWR
UMCS	PO	PW	PW	UO	UW	PG	UJ	NLU	UMK

[3]Perhaps because one of UWR's teams won the previous edition of AMPPZ.

Problem J. Jury of AMPPZ

Time limit:	1 second
Memory limit:	1024 megabytes

The AMPPZ jury consists of five people who are responsible for creating an interesting and balanced contest. This year, n problem proposals were submitted and k of them must be selected. Each juror independently orders the problems by subjective coolness and assigns them scores that form a *permutation*[4] of the numbers from 1 to n. Then the contest automatically selects the k problems with the highest average score (arithmetic mean). In case of ties, problems with smaller indices are preferred.

Four jurors have already graded all the problems. The last one is juror Jerzy, who secretly likes computational geometry and wants AMPPZ participants to experience as much of it as possible. For each problem, we know whether it is geometry ($x_i \in \{0, 1\}$), and we know the scores from the remaining jurors a_i, b_i, c_i, d_i. If Jerzy assigns his scores optimally (**also a permutation** of the numbers from 1 to n), what is the maximum possible number of selected geometry problems?

Input

The first line contains two integers n and k ($1 \le k \le n \le 200\,000$), denoting respectively the number of submitted problems and the number of problems to be selected for the contest.

The next n lines describe the submitted problems; the i-th of them contains five integers x_i, a_i, b_i, c_i, d_i ($0 \le x_i \le 1$; $1 \le a_i, b_i, c_i, d_i \le n$), describing the i-th problem. The value x_i is equal to 1 if the problem is geometry-themed, and 0 otherwise. The next four numbers are the scores given by the remaining jurors. Each juror assigned n pairwise distinct scores, so in the input each column except the first is a permutation of the numbers from 1 to n.

Output

Output a single integer – the maximum possible number of selected geometry problems.

Examples

standard input	standard output

5 3
1 3 5 5 2
0 5 2 4 5
1 2 3 3 4
1 1 1 1 1
0 4 4 2 3
	2

4 3
1 1 4 1 2
1 2 3 2 1
0 3 2 3 3
0 4 1 4 4
	1

Note

In the first example, the jury selects three of the five submitted problems. Problems 1, 3, and 4 are about geometry. Jerzy can assign the scores $(4, 5, 3, 1, 2)$ to the five problems:

1. $x_1 = 1$; average $\frac{3+5+5+2+4}{5} = 3.8$
2. $x_2 = 0$; average $\frac{5+2+4+5+5}{5} = 4.2$

[4]A sequence in any order, without repetitions — for example $(5, 4, 3, 2, 1)$ or $(2, 4, 1, 5, 3)$.

3. $x_3 = 1$; average $\frac{2+3+3+4+3}{5} = 3$
4. $x_4 = 1$; average $\frac{1+1+1+1+1}{5} = 1$
5. $x_5 = 0$; average $\frac{4+4+2+3+2}{5} = 3$

Problems 1, 2, and 3 will be selected for the contest (problems 3 and 5 have the same average, but problem 3 has the smaller index). Two of the three selected problems are geometry problems. Jerzy cannot get more than two geometry problems selected, so the answer is 2.

In the second example, Jerzy cannot get both geometry problems selected. They would both have to receive the highest score 4, which is impossible because Jerzy's scores must be pairwise distinct.

Problem K. Kids' Blocks

Time limit:	3 seconds
Memory limit:	1024 megabytes

Together with little Kostek, you are building towers out of blocks. You have already built n towers in a row, the i-th of height h_i. Now Kostek wants to set a boundary between his and your towers. The towers to the left of the boundary will belong to Kostek, and the remaining towers on the right will be yours. Each of you must have at least one tower.

You are about to move on to the next game, which begins with each of you placing a scout on your highest tower. The game won't be fun if the scouts are standing on completely different heights. Therefore, you should suggest the fairest possible division – choose the boundary so as to minimize the absolute value of the difference between the height of Kostek's highest tower and yours. Print this minimal difference.

Input

The first line contains an integer n ($2 \le n \le 300\,000$), denoting the number of towers.

The second line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$), denoting the heights of consecutive towers from left to right.

Output

Print one integer – the smallest possible difference in height between the tallest of your towers and the tallest of Kostek's towers.

Examples

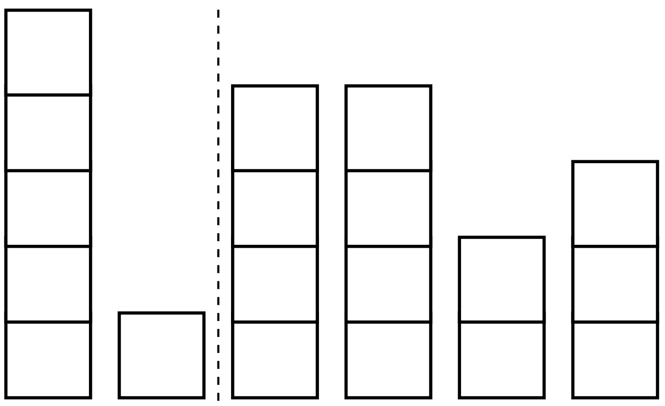
standard input	standard output
3 200 333 100	133
5 5 2 4 1 5	0
6 5 1 4 4 2 3	1

Note

In the first test, Kostek should set the boundary between the first two towers. He then has a tower of height 200, while you have towers of heights 333 and 100. Your tallest towers are 200 and 333, so the result is 133. It's not possible to achieve a smaller difference.

In the second test, any division results in a difference of 0.

In the third test, Kostek should set the boundary after the first, second, or third tower. The required difference is then $|5 - 4| = 1$. The diagram below shows one of these optimal possibilities:



Problem L. Linear Averaging

Time limit:	4 seconds
Memory limit:	1024 megabytes

You are given integers n, m, k and a sequence consisting of n integers $a_i \in [1, m]$. You can **at most once** choose an interval of the sequence (a contiguous subsequence) of length k and *average it*, that is, replace the elements of this interval with their arithmetic mean. This process takes 1 unit of time, and each element of the interval changes uniformly with a constant speed. For example, the interval with elements $(3, 9, 5, 1)$ has an average of $\frac{3+9+5+1}{4} = 4\frac{1}{2}$, so after time $t = \frac{2}{3}$ the values are $(4, 6, 4\frac{2}{3}, 3\frac{1}{3})$, and after time $t = 1$ they become $(4\frac{1}{2}, 4\frac{1}{2}, 4\frac{1}{2}, 4\frac{1}{2})$. Formally, after time $t \in [0, 1]$, an element with initial value a equals $a' = \bar{a} \cdot t + a \cdot (1 - t)$, where $\bar{a}$ is the mean of the interval. Only the elements in the chosen interval change.

Independently for each number x from 1 to m, your task is to determine how quickly the value x can appear anywhere in the sequence, by performing the process of averaging some interval. After how much time is this possible? Print `-1` if it is not possible to obtain the number x.

Input

The first line contains three integers n, m and k ($1 \le k \le n \le 300\,000$; $1 \le m \le 500\,000$), denoting respectively the length of the sequence, the upper limit of values, and the length of the modified interval.

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

Output

Print m lines. In the x-th line, print the minimum required time (a real number) needed to obtain the number x, or print the integer `-1` (without a dot) if it is not possible using at most one averaging operation.

The acceptable absolute error is 10^{-9}. You may print up to 20 digits after the decimal point.

Examples

standard input	standard output
8 10 4 3 9 5 1 1 1 1 10	0.0000000000 0.2857142857 0.0000000000 0.3333333333 0.0000000000 0.5925925926 0.4000000000 0.2000000000 0.0000000000 0.0000000000
1 3 1 2	-1 0.0000000000 -1

Note

In the first test, we can average one interval of length $k = 4$. The exact results are $0, \frac{2}{7}, 0, \frac{1}{3}, 0, \frac{16}{27}, \frac{2}{5}, \frac{1}{5}, 0, 0$.

The number $x = 6$ can be obtained the earliest at time $t = \frac{16}{27} \approx 0.5926$, by averaging the last four numbers $(1, 1, 1, 10)$ to the mean $\frac{13}{4} = 3\frac{1}{4}$. The last element is then equal to $3\frac{1}{4} \cdot \frac{16}{27} + 10 \cdot \frac{11}{27} = 6$.

The number $x = 7$ can be obtained the earliest at time $t = \frac{2}{5}$, by averaging the interval $(9, 5, 1, 1)$ to the mean 4. The first of these elements is then equal to $4 \cdot \frac{2}{5} + 9 \cdot \frac{3}{5} = 7$.

In the second test, averaging a single element does not change its value. The result for $x = 2$ is $t = 0$, whereas the numbers 1 and 3 are not reachable.

Problem M. Market

Time limit:	2 seconds
Memory limit:	1024 megabytes

Speculation in the real estate market is stressful work with unstable earnings. Would it help if you got a huge loan from your parents and could precisely predict apartment prices on selected days?

You start without an apartment, but your bank balance is 10^{100}. Since you lack a real estate agent license, you may own at most one apartment at a time. You have already spotted a specific apartment in a good location, and that is the one you may buy and then sell, possibly multiple times.

The industry portal MyHouse has published predicted intervals $[a_i, b_i]$ (with $a_i < b_i$) of the apartment's price for the next n days. The price of the apartment on the i-th day is a random **real number** c_i chosen uniformly at random from the interval $[a_i, b_i]$. By default, you learn the price c_i at the beginning of the i-th day. You must then decide whether to make a transaction. If you own the apartment, you may sell it for c_i, and if you do not own it, you may buy it for c_i.

Only today (before the first day) the MyHouse portal allows new users to learn the exact prices c_i for a chosen set of k days. That is, you now choose a set of k days and learn their prices, while for each of the remaining $n - k$ days you will learn the price at the beginning of that day.

After n days you must end without an apartment; your profit is the difference between your final and initial bank balance. Determine the maximum expected value[5] of the profit, if you optimally choose the set of k days and then optimally make transaction decisions on each of the n days.

Input

The first line contains two integers n and k ($0 \le k \le n \le 300\,000$; $1 \le n$), representing respectively the number of days and the number of prices you may learn immediately.

The next n lines describe price intervals; i-th line contains two integers a_i and b_i ($1 \le a_i < b_i \le 10^6$).

Output

Print one real number – the maximum possible expected value of the profit after n days.

The acceptable relative or absolute error is 10^{-6}. That is, if you print x and the correct exact result is y, then it must hold that $|x - y| \le 10^{-6} \cdot \max(1, y)$. You may print up to 20 digits after the decimal point.

[5]The expected value is the average value, weighted by probability, of a random variable. Intuitively, it is the result you would expect on average when repeating a random experiment many times.

Examples

standard input	standard output

4 0
10 30
10 30
40 50
30 60
	28.75

4 1
10 30
10 30
40 50
30 60
	31.3888888889

6 3
10 50
30 70
50 70
30 50
30 40
10 40
	33.6805555556

Note

In the first test we have $k = 0$, so each price is learned at the beginning of its day. There exists an optimal strategy of buying the apartment on day 1 or 2 at an expected cost of 17.5, and then selling it on day 3 or 4 with an expected income of 46.25. The expected profit is $46.25 - 17.5 = 28.75$.

In the second test we have $k = 1$ and we should optimally choose day 4, so we immediately learn a real number c_4 from the interval $[30, 60]$. The expected profit is $31\frac{7}{18}$ for optimal transaction decisions.

In the third test the optimal choice of the set of k days is $\{2, 3, 6\}$. We immediately learn the prices c_2, c_3, and c_6.