AUST CSE Carnival 1.0 Programming Contest - Preliminary (Senior 1)

https://toph.co/contests/training/p3vzxqc

Toph

Schedule

The contest will run for **3h0m0s**.

The standings will be frozen for the last **30m0s** of the contest.

Authors

The authors of this contest are IbnaHamza, ineffablekenobi, nirjoydebnath, and sohomsahaun.

Rules

You can use Bash 5.0, Brainf*ck, C# Mono 6.0, C++11 GCC 7.4, C++14 GCC 8.3, C++17 GCC 9.2, C++20 GCC 12.1, C11 GCC 12.1, C11 GCC 9.2, Common Lisp SBCL 2.0, Erlang 22.3, Free Pascal 3.0, Go 1.18, Grep 3.7, Haskell 8.6, Java 1.8, Kotlin 1.1, Lua 5.4, Node.js 10.16, Perl 5.30, PHP 7.2, PyPy 7.1 (2.7), PyPy 7.1 (3.6), Python 2.7, Python 3.7, Ruby 2.6, Rust 1.57, Swift 5.3, and Whitespace in this contest.

Be fair, be honest. Plagiarism will result in disqualification. Judges' decisions will be final.

Notes

There are 8 challenges in this contest.

Please make sure this booklet contains all of the pages.

If you find any discrepencies between the printed copy and the problem statements in Toph Arena, please rely on the later.

Disclaimer

The contents of this contest have not been reviewed by Toph and do not necessarily represent Toph's views.

A. Do You Have Paper?

You have given the responsibility to decorate the department for the upcoming AUST CSE Carnival 1.0. There are paper cuttings of the letters of the English alphabet, **in small cases**. You have to write the names of the batches using them.

The task is almost done, and you only have N letters left. You are wondering if you can write **"assembler"** or **"operand"** using the letters.

Input

First line of the input contains the number of test cases $T(1 <= T <= 100)$. Then $2T$ lines follow.

Each test case contains a number $N(1 <= N <= 100)$ denoting the number of letters and a string $S(|S| = N)$ denoting the letters you have remaining.

S only contains **small** letters of the English alphabet.

Output

For each test case, print **"Yes"** if you can write "assembler" or "operand" using the given letters. Print **"No"** otherwise.

Samples

Input	Output
3 9 assembler 8 operator 26 abcdefghijklmnopqrstuvwxyz	Yes No Yes

For the first test case, you can make "assembler" using all letters.

For the second test case, you can not make "assembler" or "operand" by rearranging the letters in any way.

For the third test case, you can take letters at positions 15, 16, 5, 18, 1, 14, 4 and make "operand".

B. Painting a Red X

The ground floor of AUST has been renovated. It now looks like a square with tiles. The tiles are painted white.

Petya thinks it doesn't look good without a red X somewhere on the ground floor. So he decides to paint a big red X at the tile centered at (R, C) by coloring the tiles. He initially wanted to make the sides of the X **three tiles wide**. To gain more visibility from above, he decided to **not color** the tiles at the center of the sides of X. (See sample tests for clarity.)

Input

The only line in the input contains three integers: $N (1 <= N <= 1000)$, $R (1 <= R <= N)$ and $C (1 <= C <= N)$.

Output

Print the ground floor after drawing the X.

White color is denoted as a **dot (.)**.

Red color is denoted as a **star (*)**.

Samples

Input	Output

`11 6 6`	`.*.......*.`
	`*.*.....*.*`
	`.*.*...*.*.`
	`..*.*.*.*..`
	`...*.*.*...`
	`....*.*....`
	`...*.*.*...`
	`..*.*.*.*..`
	`.*.*...*.*.`
	`*.*.....*.*`
	`.*.......*.`

Input	Output
7 3 7	...*.*. ....*.* .....*. ....*.* ...*.*. ..*.*.. .*.*...

C. Nodes Are in Debt

There is a directed graph with **n** nodes. Every node has a unique name from 1 to **n**. Every node has a weight $w_1, w_2, w_3 \ldots, w_n$. There are **m** directed paths from x_i to y_i for all $(1 \leq i \leq m)$. The limit of outgoing path of a node is at most 1. Here you can perform two kind of operation as many times you want:

- Choose a directed path from node **x** to **y**. Transfer some weight from **x** to **y** (The weight of **x** must be non negative after transfer).
- Choose a node **x**. Remove some weight from **x.** The weight of **x** must be non-negative after removal.

Your goal is to make the weights of all nodes equal after performing these operations any number of times (possibly zero). What is the maximum weight you can get?

Note that, there will be no loop, no multiple edges between to nodes, no self loop in the given graph.

Input

The first line contains two integers $n, m (1 \leq n \leq 10^5, 0 \leq m < n)$.

Next line contains n integers $w_1, w_2, w_3 \ldots, w_n (1 \leq w_i \leq 10^9)$.

Next m line contains x_i and y_i denoting a path from x_i to $y_i (1 \leq x_i, y_i \leq n)$.

Output

Output the maximum equal weight can be gained after performing the operations in one line.

Samples

Input	Output
4 2 2 4 2 1 2 3 3 4	2

<table>
<tr><th><u>Input</u></th><th><u>Output</u></th></tr>
<tr><td colspan="2">Following operations were executed:

1. Chooses node 2 and 3. Transfer 2 weight to node 3. The weights of all nodes are [2,2,4,1].
2. Chooses node 3 and 4. Transfer 2 weight to node 4. The weights of all nodes are [2,2,2,3].
3. Chooses node 4. Remove 1 weight from node 4. The weights of all nodes are [2,2,2,2].

After performing the operations, all nodes are equal with their nodes. You cannot obtain a greater weight than 2.</td></tr>
</table>

D. Waiting for Zeroes

Bob saved up for 5 years for his favorite toy, the **2X-er**. The machine takes a number as input and gives back its double. For example, if you give the machine 10, it gives you 20. If you give it 43, it gives you 86.

Bob is keen on wasting his free time. He takes an integer N and inserts it into the 2X-er. He takes the output and inserts it into the machine again. He takes the new output and inserts it into the machine again. This is so fun that we wants to keep doing it forever! When asked, Bob says he wants to check how many trailing zeroes he can get from N using the 2X-er.

For example, if Bob has the number 15, he puts it in the 2X-er. It gives out 30. He takes 30 and inserts it into the 2X-er again. He gets 60, then 120, 240, 480, … so on. It can be shown that Bob can never get more than 1 trailing zero.

Input

First line of the input contains the number of test cases $T(1 <= T <= 10^5)$. Then T lines follow.

Each test case contains a number $N(1 <= N <= 10^{18})$ denoting the number Bob has initially.

Output

Print the number of trailing zeroes Bob can get if he continues to use the 2X-er forever.

Samples

Input	Output
2 15 300	1 2

The first test case is explained in the statement.

For the second test case, Bobs operations look like the following:

300*2 = 600

600*2 = 1200

1200*2 = 2400

2400*2 = 4800

…

It can be proved that Bob will get no more than 2 trailing zeroes.

E. Bored Students

The students of WeirdLand got bored and decided to enter the anime world. After working for several months, they were finally able to make a portal using which they can enter the anime world and placed it in a station.

There are N students, each having a **distinct** id from 1 to N. In order for it to work correctly, they have to enter the portal in the ascending order of their id. That is, if there are 4 students, the student with id 1 will enter first, then the student with id 2, then id 3 and finally id 4. The students arrive at the station in a random order, given by an array A. It indicates that student with id A_1 arrives first, then A_2, A_3, ..., A_N. There is a waiting room at the station, which can accommodate K students at a time. If a student arrives at the station and it's not their turn to enter the portal, they head towards the waiting room. They can wait in the waiting room if there is less than K students there already. Otherwise, they get angry and damage the portal. As a result, the remaining students (including them) can not use the portal anymore.

A waiting student enters the portal **when his turn comes, before the next student arrives**. You have to find the number of students who could successfully enter the anime world.

Input

The first line of the input contains an integer T $(1 \le T \le 10^4)$ — the number of test cases.

The first line of each test case contains two integers N $(1 \le N \le 10^5)$ and K $(1 \le K \le 10^5)$.

The second line of each test case contains N integers $A_1, A_2, \ldots, A_N$ $(1 \le A_i \le N)$.

It is guaranteed that the sum of N over all test cases does not exceed 10^5.

Output

For each of the T test cases:

- print a single integer, the number of students who can successfully enter anime world

Samples

Input	Output
1 10 2 9 2 1 7 3 10 8 4 6 5	3

In the first test case:

- 9th student arrives in the station. But its not his turn to enter the portal. So he waits in the rest room. Now the number of space left in the rest room is 1.
- 2nd student arrives in the station. But its not his turn to enter the portal. So he waits in the rest room. Now the number of space left in the rest room is 0.
- 1st student arrives in the station. Its his turn to enter the portal so he enters the portal and gets transferred to the anime world.
- Now its the 2nd students turn to enter. Since he is in the rest room he enters the portal before the next student arrives. Now the number of space left in the rest room is 1.
- 7th student arrives in the station. But its not his turn to enter the portal. So he waits in the rest room. Now the number of space left in the rest room is 0.
- 3rd student arrives in the station. Its his turn to enter the portal so he enters the portal and gets transferred to the anime world.
- 10th student arrives in the station. But its not his turn to enter the portal. There is no space left in the rest room either. So he gets angry and damages the portal. As a result the remaining students including him cannot enter the portal.

So the answer is 3

F. Plaantik

“Argentina champions of the world and the nation will tango all night long”

Well, I was talking about the winner of World Cup 2022. The match result was decided by penalty shootout. It was thrilling, right? But we will make penalty shootout more thrilling today. In football, five penalties are given to each team to break the tie. Penalties are taken one after the other. **Team A** takes one penalty after that **Team B** is given the opportunity to take one penalty. That is how the two teams take 5 penalties between them. The team which scores more goals are declared winner. If the scores are level then it goes to next step. Sometimes there is no need to take five penalties because the result can be decided before that. For example, If one team scores 4 goals by taking 4 kicks and another team scores only 1 by taking 3 kicks, first team will be declared as winner. In this circumstances, if the second team takes 2 of their remaining penalty and scores, the first team will still remain ahead of them. Because of this the first team is declared winner immediately.

In this problem we will change some rules. There will be **n** penalty kicks and every penalty will have a_i point for $(1 \leq i \leq n)$. If a team scores in the i_{th} penalty kick, they will receive a_i point. You will be given two strings **F** and **S** of **n** characters containing the characters ‘S’ and ‘M’. ‘S’ means the penalty was scored and ‘M’ means the penalty was missed. The team with more point will declare winner. You have to determine the minimum number of penalty needs to take by both team to declare the winner. **Team A** goes first. Penalties will be taken one after the other here also.

Input

The first line contains an integer $T(1 \leq T \leq 10)$, the number of penalty will be taken.

Next line contains an integer $n(1 \leq n \leq 10^5)$, the number of penalty will be taken.

Next line contains n integers $a_1, a_2 \ldots, a_n(1 \leq a_i \leq 10^9)$, points of each penalty kick.

Next two lines will contain two strings **F** and **S**.

The sum of **n** all over testcases will not exceed 100000.

Output

For each test case, Print “**Draw**” if both team gets same point after all penalty kicks otherwise print two integers **x** and **y** denoting the minimum penalty kicks were needed by the **Team A** and by the **Team B** to declare the winner.

Samples

Input	Output
3 4 1 1 1 1 SSSS MMSS 3 1 1 100 SSM MMS 5 1 1 1 1 1 SSSSS SSSSS	3 2 3 3 Draw

In the first test case,

Team A scores their first penalty kick. They have got 1 point and **Team B** got 0. **Team B** is behind but they have 1+1+1+1=4 more points to play, so they have the chance to **win** if **Team A** misses some penalty kicks.

Team B misses their first penalty kick. They have got 0 point and **Team A** got 1. **Team B** is behind but they have 1+1+1=3 more points to play, so they have the chance to **win** if **Team A** misses some penalty kicks.

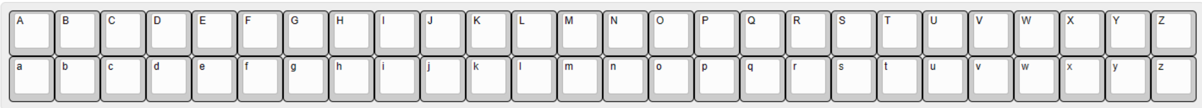
Team A scores their second penalty kick. They have got 2 point and **Team B** got 0. **Team B** is behind but they have 1+1+1=3 more points to play, so they have the chance to **win** if **Team A** misses some penalty kicks.

Team B misses their second penalty kick. They have got 0 point and **Team A** got 2. **Team B** is behind but they have 1+1=2 more points to play, so they have the chance to **draw** if **Team A** misses every next penalty kicks.

Team A scores their second penalty kick. They have got 3 point and **Team B** got 0. **Team B** is behind and they have 1+1=2 more points to play, if **Team B** scores both of their penalty kicks and **Team A** misses their last one, **Team B** will lose anyway. So the penalty shootout ends here after **Team A** takes 3 penalty kicks and **Team B** takes 2 penalty kicks.

G. Lazy Leo's Password

Leo has a keyboard with 2 rows and 52 keys. The first row contains 26 keys, denoting the capital letters of the English alphabet $(A - Z)$. The second row contains 26 keys, denoting the small letters of the English alphabet $(a - z)$.



Leo has to come up with a password for his new social media account. He is wondering how many different passwords of length L he can make. Since Leo is very lazy, he always presses the same key, or a key adjacent to the previous key he pressed. For example, if he presses D, he presses one of D, C, E or d in his next move. If he presses Z, he presses one of Z, Y or z in his next move.

Obviously, Leo does not want to do this task himself. He asks you how many different passwords of length L he can make.

Input

The first line contains an integer Q $(1 \le Q \le 2 * 10^5)$, the number of queries by Leo.

Each of the next Q lines contains a single integer L $(1 \le L \le 2 * 10^5)$, length of the password Leo wants to make.

Output

For each query, print the number of different passwords Leo can make. Since the result is very large, print the remainder of the result when divided by 1000000007.

Samples

Input	Output
3 1 2 1000	52 204 720235484

H. Precision Problems

Precision is a big problem when it comes to dealing with numbers. We always want a solution that is precise. We'll be given a decimal number n and find a number x such that if we subtract n% of x the answer would be n.

Input

The first line of input contains an integer t (1 <= t <= 10^5)

The next t lines will contain a decimal number n (1 <= n < 100)

Output

Print a number x such that if we subtract n% of x the answer would be n.

Error less than 10^-6 will be ignored

Samples

Input	Output
5 45 80 25 95 37	81.818182 400.000000 33.333333 1900.000000 58.730159