Kattis

Problem Solving on your Own

2025-03-14

Problems

Advice, hints, and general information

- Your solution programs should read input from standard input (e.g. System.in in Java or std::cin in C++) and produce output on standard output (e.g. System.out in Java or std::cout in C++). Anything written on standard error will be ignored. For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is “No comment, read problem statement”, indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem.

Pie

CPU TIME LIMIT

2 seconds

MEMORY LIMIT

1024 MB

My birthday is coming up and traditionally I'm serving pie. Not just one pie, no, I have a number N of them, of various tastes and of various sizes. F of my friends are coming to my party and each of them gets a piece of pie. This should be one piece of one pie, not several small pieces since that looks messy. This piece can be one whole pie though.
My friends are very annoying and if one of them gets a bigger piece than the others, they start complaining. Therefore all of them should get equally sized (but not necessarily equally shaped) pieces, even if this leads to some pie getting spoiled (which is better than spoiling the party). Of course, I want a piece of pie for myself too, and that piece should also be of the same size.

What is the largest possible piece size all of us can get? All the pies are cylindrical in shape and they all have the same height 1, but the radii of the pies can be different.

Input

One line with a positive integer: the number of test cases (at most 25). Then for each test case:

- One line with two integers N and F with $1 \leq N, F \leq 10\,000$: the number of pies and the number of friends.
- One line with N integers r_i with $1 \leq r_i \leq 10\,000$: the radii of the pies.

Output

For each test case, output one line with the largest possible volume V such that me and my friends can all get a pie piece of size V. The answer should be given as a real number with an absolute error of at most 10^{-3}.

Sample Input 1

```
3
3 3
4 3 3
1 24
5
10 5
1 4 2 3 4 5 6 5 4 2
```

Sample Output 1

```
25.132741229
3.141592654
50.265482457
```

Report Card

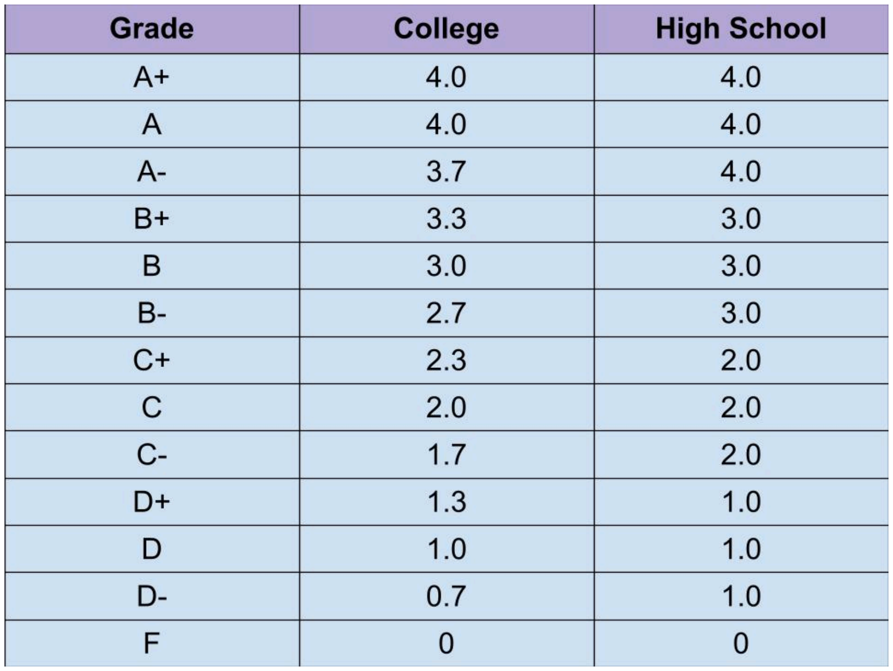
CPU TIME LIMIT

4 seconds

MEMORY LIMIT

1024 MB

After coming to college from high school, you discovered something absolutely tragic. The way that GPA is calculated is completely different than what you are used to. Apparently, pluses and minuses are treated very differently than they were before, according to the following table.



Grade	College	High School
A+	4.0	4.0
A	4.0	4.0
A-	3.7	4.0
B+	3.3	3.0
B	3.0	3.0
B-	2.7	3.0
C+	2.3	2.0
C	2.0	2.0
C-	1.7	2.0
D+	1.3	1.0
D	1.0	1.0
D-	0.7	1.0
F	0	0

Figure 1: Grade Point Table Comparing High School Against UCLA

GPA is calculated by taking the sum of the individual grade points (seen in the table above) and dividing it by the number of courses. You have received your report card for the current quarter, but you've blocked out the pluses and minuses instead because you're too scared to look, so your report card can be one of many different variations. For example, a report card of ABB could be A+ B B+, A− B− B+, or even A B B. In how many possible report card variations will the college grading system actually be better for your GPA compared to your old high school system? Since this number might be quite large, print out your answer modulo $10^9 + 7$. Note that order matters, so `A B B+` and `A B+ B` are considered two different variations of ABB.

Input

The input starts with one line containing an integer N, denoting the length of the report card ($1 \leq N \leq 3\,000$).

The second line is a string of N characters (A,B,C,D,F) representing the obscured report card.

Output

The number of report cards modulo $10^9 + 7$ that would give you a strictly better GPA than in high school.

Sample Input 1

```
2
AB
```

Sample Output 1

```
2
```

Sample Input 2

```
3
CCF
```

Sample Output 2

```
3
```

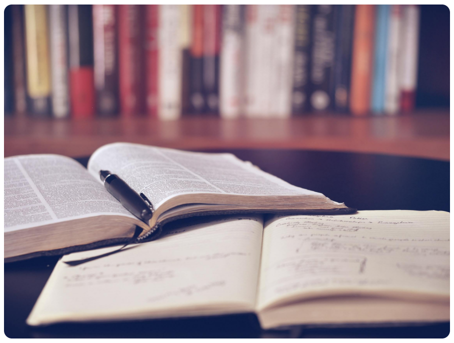
H-Index

CPU TIME LIMIT

1 second

MEMORY LIMIT

1024 MB



In research, it is tough to determine how good of a researcher you are. One way that people determine how good you are is by looking at your *H-Index*.

Each paper has a certain number of citations. Your H-Index is the largest number H such that you have H papers with at least H citations. Given the number of citations on each paper you have written, what is your H-Index?

Input

The first line of input contains a single integer n ($1 \leq n \leq 100\,000$), which is the number of papers you have written.

The next n lines describe the papers. Each of these lines contains a single integer c ($0 \leq c \leq 1\,000\,000\,000$), which is the number of citations that this paper has.

Output

Display your H-Index.

Sample Input 1

```
5
7
1
2
1
5
```

Sample Output 1

```
2
```

Sample Input 2

```
5
7
1
3
1
5
```

Sample Output 2

```
3
```

Sample Input 3

```
3
4
2
3
```

Sample Output 3

```
2
```

Santa Klas

CPU TIME LIMIT

1 second

MEMORY LIMIT

1024 MB

Picture by m.prinke

Santa Klas is in trouble! The ACM (Aerodynamic Christmas Machines) has called you in to be their subject matter expert for the current situation.

Santa Klas recently got himself a new bunch of reindeers, that would boost his speed even more while travelling across the sky. The reindeers even came with an autopilot, so Santa Klas calmly took off into the sky, adjusted his heading, activated his autopilot and fell asleep.

Santa Klas has been playing too much flight simulator computer games lately, and he is certain that when the autopilot is activated, the ship he is flying in will hold altitude and course. This is not the case. The autopilot on his new reindeers only guarantee that the heading is the same, and the vertical angle is kept the same. This means that if Santa Klas' ship is currently heading for the ground, he will crash if he doesn't wake up soon enough! If the ship isn't pointing to the ground, Santa Klas will exit the atmosphere. But that's fine, he's done that before.

The ACM has given you Santa Klas' angle between his course and the ground and altitude in meters. Help them determine how much time they have to wake him up, or if Santa Klas will exit the atmosphere. Santa Klas always flies with a speed of one meter per second (yeah, it takes him a while to round the earth).

Input

The only line of input contains two integers $1 \leq H \leq 10\,000, 0 \leq v \leq 359$, Santa Klas' current altitude and his current angle in degrees from the horizon, i.e. if $v = 0$ then Santa

Klas is flying with constant altitude (since the earth is flat), and is therefore safe. If $v = 90$, then Santa Klas is heading straight for the stars.

Output

Output should consist of one single integer: the number of whole seconds that the ACM have to safely wake Santa Klas up, so that he can adjust his course. You may assume that Santa Klas is infinitely fast at adjusting it after waking up, as the flight simulator professional he is. If Santa Klas is not heading for the ground, just print the word "`safe`" on a single line instead.

Sample Input 1

```
30 270
```

Sample Output 1

```
30
```

Sample Input 2

```
1 180
```

Sample Output 2

```
safe
```

Progressive Scramble

CPU TIME LIMIT

1 second

MEMORY LIMIT

1024 MB

Photo by Chilanga Cement

You are a member of a naive spy agency. For secure communication, members of the agency use a very simple encryption algorithm – which changes each symbol in the message 'progressively', i.e., based on the symbols preceding it. The allowed symbols are space and the 26 lowercase English letters. For encryption purposes we assign them the values 0 (for space) and 1 through 26 (for a–z). We'll let $v(s)$ represent the numeric value of symbol s.

Consider a message with symbols $s_1, s_2, \dots, s_n$. The encryption algorithm starts by converting the first symbol s_1 into its associated value $u_1 = v(s_1)$. Then for each subsequent symbol s_i in the message, the computed value is $u_i = v(s_i) + u_{i-1}$ — the sum of its associated value and the computed value for the previous symbol. (Note that when there is a space in the input message, the previous scrambled letter is repeated.) This process continues until all the u_i are computed.

At this point, the message is a sequence of numeric values. We need to convert it back to symbols to print it out. We do this by taking the value u_i modulo 27 (since there are 27 valid symbols), and replacing that value with its corresponding symbol. For example, if $u_i = 32$, then $32 \bmod 27 = 5$, which is the symbol 'e' (since $v(e) = 5$).

Let's look at an example. Suppose we want to encrypt the string "my pie".

1. First, convert each symbol s_i into $v(s_i)$: $[13, 25, 0, 16, 9, 5]$.
2. Next, compute each u_i: $[13, 38, 38, 54, 63, 68]$.
3. Then, use modulus on the u_i: $[13, 11, 11, 0, 9, 14]$.

4. Finally, convert these back to symbols: “mkk in”.

Create a program that takes text and encrypts it using this algorithm, and also decrypts text that has been encrypted with this algorithm.

Input

The input to your program consists of a single integer $1 \le n \le 100$ on its own line. This number is followed by n lines, each containing the letter ‘e’ or ‘d’, a single space, and then a message made up of lowercase letters (a–z) and spaces, continuing to the end of the line. Each message is between 1 and 80 characters long. The letters ‘d’ and ‘e’ indicate that your program decrypts or encrypts the subsequent string, respectively.

Output

Output the result of encrypting or decrypting each message from the input on its own separate line. Note that differences in whitespace are significant in this problem. Therefore your output must match the correct output character-for-character, including spaces.

Sample Input 1

```
7
e testing multiple letters
rrrrrrrrrrrrr
e this particularly long sentence
can test encryption
d
tajbbrsjcloiuvmywwhwjqqqinauzmpuux
yllejbvv nqhfvoxlz
e my pie
d mkk in
e the quick brown fox jumps over
the lazy dog
d taffwqzbmmofuqddjyvvezlatthchzzs
eeqrqoosgn
```

Sample Output 1

```
tyqjsfmmzteygwhmmycwpulddvmdvmdvmd
vmdv
tajbbrsjcloiuvmywwhwjqqqinauzmpuux
yllejbvv nqhfvoxlz
this particularly long sentence
can test encryption
mkk in
my pie
taffwqzbmmofuqddjyvvezlatthchzzs
eeqrqoosgn
the quick brown fox jumps over the
lazy dog
```

Let's Play Monopoly!

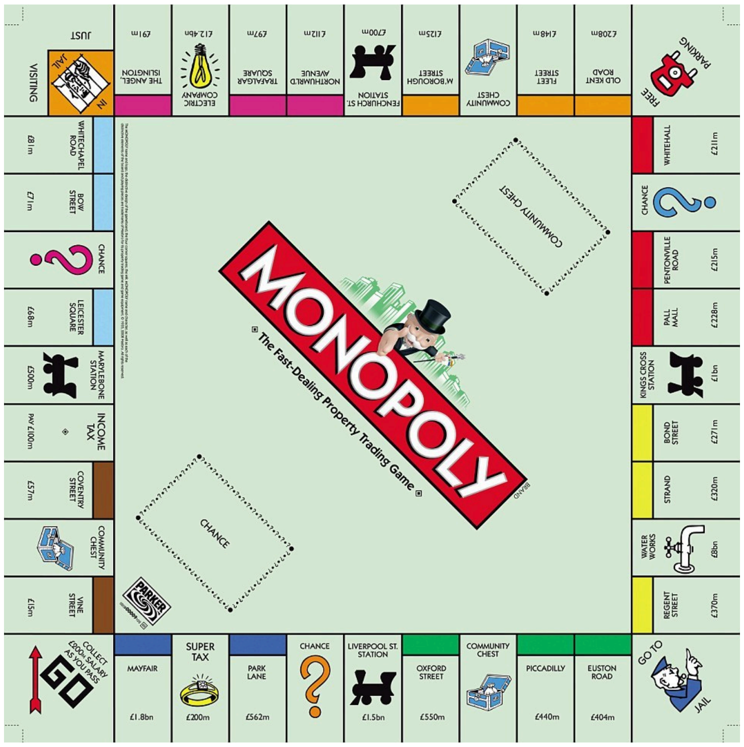
CPU TIME LIMIT

1 second

MEMORY LIMIT

256 MB

Monopoly is a fast-dealing property trading game, which is firstly designed by Elizabeth Magie and then published as well as owned by Hasbro. Since its first appearance in 1935, this game has quickly become well-known and deserves the best way of enjoying with friends at leisure. In this game, you can demonstrate your powerful investing skill and your great support from the God by throwing dices, travelling on a risky board and building your real estate for the purpose of forcing your opponents to pay huge amounts of money for renting your land, which eventually leads to bankruptcy. Various localized versions of Monopoly have been introduced, consisting of different famous places all over the world, while their layout share the same. The figure below shows the board of *UK Edition Monopoly*.



If you have enjoyed delightful moments with *Monopoly*, I wish you can entertain yourselves by solving this problem. Otherwise, do not worry as in this problem we are playing a significantly simplified version of this game. Forget the rules of the traditional game and read carefully the below description:

In this problem, instead of a board, the game is held in a graph with N nodes and M directed edges. Two players, Alob and Bice join the game. At the beginning, Alob is at node s_a while Bice is at node s_b. Alob starts first. Two players take turn alternatively. In each turn, a player has to move from his current node, along some edge to another node; and may pay, collect money or buy property, depending on the type of the new node. If he can not choose any edges to pass through, he has to pass that turn (do nothing).

Every node belongs to one of the following types:

- **Property**: You can purchase to own this property and force your opponents to pay you when they reach here. Each property has its own *buying cost* and *renting cost*. Initially, all properties are unowned. When you reach an unowned property, you can decide either to buy this property (you have to pay its *buying cost*) to **permanently** own it, or to ignore. After you purchase for this property, your opponent has to send you an amount of money equal to its *renting cost* everytime he reaches it. Of course, you have nothing to do while landing on your own properties.

- **Salary**: Each node of this kind has its own value, which is the amount of money you gain everytime you reach this.

- **Tax**: Each node of this kind has its own value, which is the amount of money you lose when you reach this node.

Since Alob and Bice are bored of playing an overlong game, each person will take at most K turns (including passed turns). After that, the game ends. They had already won the gameshow *Who Wants to Be a Millionaire?* so they never worry about running out of cash. Each player wants his money to be more than his opponent's money as much as possible. If he has several strategies resulting in the same difference, he always prefers the one giving him largest amount of money.

Assume that both player play optimally, your task is to determine the outcome of the game.

Input

The first line contains five integers: N, M, K, s_a, s_b - the number of nodes and edges in the graph, the maximum number of turns each player takes, and the starting nodes of Alob and Bice, respectively.

The i^{th} line of the next M lines contains two integers u_i and v_i, representing an edge from u_i to v_i.

The i^{th} line of the last N lines describes the i^{th} node of the graph in one of the following formats:

- *PROPERTY b r*: Denotes that this node has a property with *buying_cost* b and *renting_cost* r.
- *SALARY v*: Denotes that a player earns v when entering this node.
- *TAX v*: Denotes that a player loses v when entering this node.

Output

Write out in one line two space-separated integers denoting the amount of money Alob and Bice gain at the end of the game, respectively.

Constraints

- $1 \leq N \leq 10^5$
- $0 \leq M \leq 10^5$
- $10^6 \leq K \leq 10^9$
- $1 \leq s_a \leq N, 1 \leq s_b \leq N$
- All other numbers in the input files are integers between 1 and 10^9, inclusive.

- For all nodes which has property, $renting\_cost \leq \frac{buying\_cost}{\pi}$.
- For every i from 1 to M, $1 \leq u_i < v_i \leq N$.

Sample Input 1

```
12 12 123456789 1 2
1 3
1 5
3 7
3 9
5 9
7 11
2 4
2 6
4 8
4 10
6 8
8 12
SALARY 1
SALARY 10000
TAX 3
TAX 200
SALARY 10
TAX 1000
SALARY 7
PROPERTY 50 14
TAX 18
PROPERTY 105 33
PROPERTY 11 2
SALARY 7
```

Sample Output 1

```
4 -193
```

Text Encryption

CPU TIME LIMIT

1 second

MEMORY LIMIT

1024 MB

To keep privacy of messages and prevent the aliens from reading them, we may use various encryption algorithms. These algorithms encode a message into the so-called *ciphertext* that is difficult (or impossible) to decode for anyone else than the intended recipient. *Transposition ciphers* are a type of encryption that do not change the letters of the message but only change their order ("shuffle" the letters). Of course, the shuffling must be reversible to allow later decryption.

In this problem, we will consider a simple transposition cipher which shuffles the letters in such a way that the *decryption* algorithm always takes every n-th letter. More specifically: when decrypting, the first letter of the ciphertext is taken first, then the next $n-1$ letters are (repeatedly) skipped and the next letter taken, and so on until we reach the end of the ciphertext. After that, we repeat the procedure starting with the second letter of the ciphertext, and so on until all letters are used.

Your task is to implement the encryption algorithm for this cipher. For a given message, produce the encrypted text (ciphertext). To make the cipher a little bit stronger, you should convert all letters to uppercase and leave out all spaces between words.

Input

The input contains at most 150 messages. Each message is described by two lines. The first line contains an integer n ($1 \leq n \leq 1\,000$). The second line contains the message. The message will be at most $10\,000$ characters long, it will only contain letters and spaces, and there will be at least one letter in each message.

The last message is followed by a line containing zero.

Output

For each message, output the ciphertext that, after using the described decryption algorithm, will result in the original message (with all spaces removed and all letters in uppercase).

Sample Input 1

```
2
CTU Open Programming Contest
7
This  is  a  secret  message  that
noone should ever see Lets encrypt
it
15
text too short
0
```

Sample Output 1

```
CMTMUIONPGECNOPNRTOEGSRTA
TESNUECHCAOLERIRGODLYSEENEEPITTEVT
TSMHSESIAEAHRETSSTOSN
TEXTTOOSHORT
```