ADA - Análisis y Diseño de Algoritmos, 2024-1
Tarea 1: Semanas 1 y 2

Para entregar el domingo 4 de febrero de 2024

Problemas conceptuales a las 23:59 por BrightSpace

Problemas prácticos a las 23:59 en la arena de programación

Tanto los ejercicios como los problemas deben ser resueltos, pero únicamente las soluciones de los problemas deben ser entregadas. La intención de los ejercicios es entrenarlo para que domine el material del curso; a pesar de que no debe entregar soluciones a los ejercicios, usted es responsable del material cubierto en ellos.

Instrucciones para la entrega

Para esta tarea y todas las tareas futuras, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja impresa entregada o en cada archivo de código (a modo de comentario). Adicionalmente, agregue la información de fecha y nombres de compañeros con los que colaboró; igualmente cite cualquier fuente de información que utilizó.

¿Cómo describir un algoritmo?

En algunos ejercicios y problemas se pide "dar un algoritmo" para resolver un problema. Una solución debe tomar la forma de un pequeño ensayo (es decir, un par de párrafos). En particular, una solución debe resumir en un párrafo el problema y cuáles son los resultados de la solución. Además, se deben incluir párrafos con la siguiente información:

- una descripción del algoritmo en castellano y, si es útil, pseudo-código;
- por lo menos un diagrama o ejemplo que muestre cómo funciona el algoritmo;
- una demostración de la corrección del algoritmo; y
- un análisis de la complejidad temporal del algoritmo.

Recuerde que su objetivo es comunicar claramente un algoritmo. Las soluciones algorítmicas correctas y descritas *claramente* recibirán alta calificación; soluciones complejas, obtusas o mal presentadas recibirán baja calificación.

Ejercicios

La siguiente colección de ejercicios, tomados del libro de Cormen et al. es para repasar y afianzar conceptos, pero no deben ser entregados como parte de la tarea.

1.2-2, 1.2-3 (página 15), 2.2-3 (página 33).

Problemas conceptuales

1. Escribir el código de honor del curso.
2. Problema 1-1: *Comparison of running times* (Cormen et al., página 15).

Problemas prácticos

Hay cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - Almost Union-Find

Source file name: `almost.py`
Time limit: x seconds

I hope you know the beautiful Union-Find structure. In this problem, you're to implement something similar, but not identical.

The data structure you need to write is also a collection of disjoint sets, supporting 3 operations:

1 p q**:** Union the sets containing p and q. If p and q are already in the same set, ignore this command.

2 p q**:** Move p to the set containing q. If p and q are already in the same set, ignore this command.

3 p**:** Return the number of elements and the sum of elements in the set containing p.

Initially, the collection contains n sets: $\{1\}$, $\{2\}$, $\{3\}$, ..., $\{n\}$.

Consider the following example with $n = 5$ sets:I

- Initially: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$.
- After operation 1 1 2: $\{1, 2\}, \{3\}, \{4\}, \{5\}$.
- After operation 2 3 4: $\{1, 2\}, \{3, 4\}, \{5\}$ (the empty set that is produced when taking out 3 from $\{3\}$ is omitted).
- After operation 1 3 5: $\{1, 2\}, \{3, 4, 5\}$.
- After operation 2 4 1: $\{1, 2, 4\}, \{3, 5\}$.

Input

There are several test cases. Each test case begins with a line containing two integers n and m ($1 \leq n, m \leq 100\,000$), the number of integers, and the number of commands. Each of the next m lines contains a command. For every operation, assume $1 \leq p, q \leq n$.

The input must be read from standard input.

Output

For each type-3 command, output 2 integers: the number of elements and the sum of elements.

The output must be written to standard output.

Sample Input	**Sample Output**

`5 7`
`1 1 2`
`2 3 4`
`1 3 5`
`3 4`
`2 4 1`
`3 4`
`3 3`
	`3 12`
	`3 7`
	`2 8`

B - Hippo Circus

Source file name: `circus.py`
Time limit: x seconds

You all know about my friend Hippo. Hippo and its other hippopotamus friends are starting a circus. They have been practicing a lot, and they are getting better at the show. I have seen their show several times and I personally think they are really good. So I encouraged them to show in public. And after a lot of arguing and convincing they finally agreed.

So are they are getting ready for their big showdown. Everything is prepared. The performers are working day and night to perfect everything. The tent is almost ready. In a word everything is having the final touch.

In the night before the show the hippopotamuses started to budget the time and encountered a big problem. They were planning for a big entrance where every hippopotamus will enter through the gate and take a bow to the audience. But this is taking too much time. So to shorten this they devised a plan – "One hippo will ride another one". The balances of the hippopotamuses are not so good yet. So a hippo can take only another hippo over it, not more than that. There is another problem, if a hippo carries another hippo, it slows the speed of the hippo. So to help them with the problem they wish your help.

Given a door with height H and N hippopotamuses with height h_i (height of the i-th hippo, $1 \leq i \leq N$), you need to find the minimum time so that every hippo can enter the door and bow. A hippo can only enter the door if its height is less than the height of the door. If a hippo is carrying another hippo, then the summation of their heights must be less than the door's height. A hippo while walking alone, takes T_a time to enter the door and bow. A hippo while carrying another hippo, takes T_d time to enter and bow.

Input

First line of input will contain an integer C ($C \geq 1$), the number of test cases. Then C cases will follow. First line of each case is four integers N, H, T_a, and T_d. Next line contains N integers, the height of the hippopotamuses. Here $1 \leq N \leq 100\,000$, and $0 \leq T_a < T_d \leq 10\,000$. All the heights will be less than 100. Heights of all the hippopotamuses will be at most H.

The input must be read from standard input.

Output

For each case output one line. '`Case X: M`' (without the quotes), where `X` is the case number starting from 1 and M is minimum time needed. Check sample input and output for details.

The output must be written to standard output.

Sample Input	**Sample Output**

`2`
`3 5 2 3`
`3 4 2`
`3 6 2 3`
`3 4 2`
	`Case 1: 6`
	`Case 2: 5`

C - Summing Digits

Source file name: `digits.py`
Time limit: x seconds

For a positive integer n, let $f(n)$ denote the sum of the digits of n when represented in base 10. It is easy to see that the sequence of numbers n, $f(n)$, $f(f(n))$, $f(f(f(n)))$, ... eventually becomes a single digit number that repeats forever. Let this single digit be denoted $g(n)$.

For example, consider $n = 1234567892$. Then:

$$\begin{aligned} f(n) &= 1+2+3+4+5+6+7+8+9+2 = 47 \\ f(f(n)) &= 4+7 = 11 \\ f(f(f(n))) &= 1+1 = 2. \end{aligned}$$

Therefore, $g(1234567892) = 2$.

Input

Each line of input contains a single positive integer n at most 2 000 000 000. Input is terminated by $n = 0$ which should not be processed.

The input must be read from standard input.

Output

For each such integer, you are to output a single line containing $g(n)$.

The output must be written to standard output.

Sample Input	**Sample Output**

`2`
`11`
`47`
`1234567892`
`0`
	`2`
	`2`
	`2`
	`2`

D - Ahoy, Pirates!

Source file name: `pirates.py`
Time limit: x seconds

In the ancient pirate ages, the Pirate Land was divided into two teams of pirates, namely, the Buccaneer and the Barbary pirates. Each pirate's team was not fixed, sometimes the opponent pirates attacked and he was taken away to the other pirate team. All on a sudden a magician appeared in the Pirate Land, where he was making transition of pirates from their team to other team at his own will. Of course, handy spells were used. The process of changing team was known as mutating. There were N pirates and all of the pirates have a unique id from 0 to $N - 1$. The great magician could mutate a bunch of pirates with consecutive id's to another one.

Suppose there were 100 pirates in the pirate land and all of them were Barbary pirates, then the magician could cast a spell to change pirates with id's from 10 to 33 to Buccaneer pirates. Then the whole pirate land would have 24 Buccaneer and 76 Barbary pirates.

The magician was very fast casting the spell. Once, God started to dislike this. God had favor for the Buccaneer pirates and God asked the magician, "Tell me, how many of the pirates of index from 2 to 30 are Buccaneer pirates?". Now the magician was puzzled as he was only efficient in casting spells, not in counting :-)

Being clever enough, the magician captured a clever man from the Earth Land. And unfortunately its you! Now you have to answer the Gods questions.

Input

The first line of input will contain number of test cases T. For each test case, the first part of the description will be of the pirate land. There could be up to N ($1 \leq N \leq 1\,024\,000$) pirates. Each pirate is either assigned to Buccaneer or Barbary Pirate. Buccaneer pirates are described by '1' (ONE) and Barbary pirates are described by '0' (ZERO). You have to build a string of the pirates description. Each case starts with an integer M ($M \leq 100$), where M pair lines follow. In each pair of lines (we call it a set), first has an integer T ($T \leq 200$) and next one has a nonempty string Pirates (consisting of 0 and 1, 0 for Barbary, 1 for Buccaneer, has maximum length of 50). For each pair concatenate the string Pirates, T times. Concatenate all the resulting M sets of strings to build the pirate description. The final concatenated string describes the pirates from index 0 to end ($N - 1$ for N pirates). Now the next part of the input will contain queries. First line of next part has an integer Q describing number of queries. Each subsequence Q ($1 \leq Q \leq 1\,000$) lines describe each query. Each query has a string F or E or I or S, and two integers, a and b denoting indexes ($a \leq b$, $0 \leq a < n$, $0 \leq b < n$, index range are inclusive). The meaning of the query string are follows:

$F\ a\ b$ means mutate the pirates from index a to b to Buccaneer Pirates.

$E\ a\ b$ means mutate the pirates from index a to b to Barbary Pirates.

$I\ a\ b$ means mutate the pirates from index a to b to inverse pirates.

$S\ a\ b$ means "God's query" asking the question: "Tell me how many Buccaneer pirates are there from index a to b?"

The input must be read from standard input.

Output

For each test print the case number as the sample output suggests. Then for each of God's query, output the query number, colon (:) and a space and the answer to the query as the sample suggest.

The output must be written to standard output.

Sample Input

```
2
2
5
10
2
1000
5
F 0 17
I 0 5
S 1 10
E 4 9
S 2 10
3
3
1
4
0
2
0
2
I 0 2
S 0 8
```

Sample Output

```
Case 1:
Q1: 5
Q2: 1
Case 2:
Q1: 0
```

E - Rockabye Tobby

Source file name: `tobby.py`
Time limit: x seconds

”Rockabye baby, don’t you cry”.

Tobby is very good at catching the ball, he loves that game so much, that one day he decided to go out and play, even though it was raining. He played for a long time and in addition to catching the ball many times, he also got a cold, poor Tobby. That is why now his mother will take care of him, Big doggie momma, singing that beautiful lullaby (rockabye) and giving him the medications in the moments that must be taken.

In the medical prescription sent by the doctor, he specifies the name of the medications and how often they should be taken. The doctor told him that if he takes the medications constantly, he will be relieved after taking k medicines. Tobby does not like being sick (in fact no one likes to be), so he promises his mother to be constant with the medicines, that is why he now wants to know what are the first k medicines that he has to take to feel better. Can you help him?

Input

Input begins with a line containing an integer T ($T \geq 1$), the number of test cases. For each test case, the medical prescription is written as follows. Each test case begins with a line containing two integers, n ($1 \leq n \leq 3 \cdot 10^3$) and k ($1 \leq k \leq 10^4$), indicating the number of medications sent by the doctor and the minimum number of medicines Tobby must take to feel better. The following n lines will be of the form, ‘name frecuency’ ($1 \leq |\text{name}| \leq 15$, $1 \leq \text{frecuency} \leq 3 \cdot 10^3$), indicating the name of the medication and how often it should be taken. The medicines are listed according to their degree of priority, i.e. the first one will be the most important medicine and the last one, the least important.

The input must be read from standard input.

Output

For each test case, the output must have k lines, each of the form, ‘`t m`’, indicating that in the moment `t` Tobby must take the medicine `m`. If there are two or more medicines that must be given at the same time `t`, they should be printed according to their priority.

The output must be written to standard output.

Sample Input	**Sample Output**

`1`
`2 5`
`Acetaminophen 20`
`Loratadine 30`
	`20 Acetaminophen`
	`30 Loratadine`
	`40 Acetaminophen`
	`60 Acetaminophen`
	`60 Loratadine`