Problem J1: Winning Score

Problem Description
You record all of the scoring activity at a basketball game. Points are scored by a 3-point shot, a 2-point field goal, or a 1-point free throw.

You know the number of each of these types of scoring for the two teams: the Apples and the Bananas. Your job is to determine which team won, or if the game ended in a tie.

Input Specification
The first three lines of input describe the scoring of the Apples, and the next three lines of input describe the scoring of the Bananas. For each team, the first line contains the number of successful 3-point shots, the second line contains the number of successful 2-point field goals, and the third line contains the number of successful 1-point free throws. Each number will be an integer between 0 and 100, inclusive.

Output Specification
The output will be a single character. If the Apples scored more points than the Bananas, output `A`. If the Bananas scored more points than the Apples, output `B`. Otherwise, output `T`, to indicate a tie.

Sample Input 1
```
10
3
7
8
9
6
```

Output for Sample Input 1
```
B
```

Explanation of Output for Sample Input 1
The Apples scored $10 \cdot 3 + 3 \cdot 2 + 7 \cdot 1 = 43$ points and the Bananas scored $8 \cdot 3 + 9 \cdot 2 + 6 \cdot 1 = 48$ points, and thus the Bananas won.

Input for Sample Input 2
```
7
3
0
6
4
1
```

La version française figure à la suite de la version anglaise.

Output for Sample Input 2

```
T
```

Explanation of Output for Sample Input 2

The Apples scored $7 \cdot 3 + 3 \cdot 2 + 0 \cdot 1 = 27$ points and the Bananas scored $6 \cdot 3 + 4 \cdot 2 + 1 \cdot 1 = 27$ points, and thus it was a tie game.

La version française figure à la suite de la version anglaise.

Problème J1 : Le pointage des vainqueurs

Énoncé du problème
Vous enregistrez le pointage lors d'un match de basket-ball entre l'équipe des Ananas et l'équipe des Bananes. Les points sont marqués par une équipe lorsqu'un joueur réussit un tir à 3 points, un tir à 2 points, ou un tir à 1 point (un lancer franc).

Vous connaissez le nombre de tirs de chaque type que les deux équipes ont marqué. Votre travail consiste à déterminer laquelle des équipes était victorieuse ou si elles ont fait match nul.

Précisions par rapport aux données d'entrée
Les trois premières lignes des données d'entrée sont relatives au pointage de l'équipe des Ananas tandis que les trois lignes d'après sont relatives au pointage de l'équipe des Bananes. Pour chaque équipe, la première ligne contient le nombre de fois qu'un tir à 3 points a été réussit, la deuxième le nombre de fois qu'un tir à 2 points a été réussit et la troisième le nombre de fois qu'un tir à 1 point a été réussit (un lancer franc). Chaque nombre sera un entier de 0 à 100.

Précisions par rapport aux données de sortie
Les données de sortie ne contiendront qu'un seul caractère. Si les Ananas ont marqué plus de points que les Bananes, la donnée de sortie devrait être « A ». Si les Bananes ont marqué plus de points que les Ananas, la donnée de sortie devrait être « B ». Finalement, la donnée de sortie d'un match nul devrait être « T ».

Données d'entrée d'un 1er exemple
```
10
3
7
8
9
6
```

Données de sortie du 1er exemple
```
B
```

Justification des données de sortie du 1er exemple
Les Ananas ont marqué $10 \cdot 3 + 3 \cdot 2 + 7 \cdot 1 = 43$ points tandis que les Bananes ont marqué $8 \cdot 3 + 9 \cdot 2 + 6 \cdot 1 = 48$ points. L'équipe des Bananes est donc victorieuse.

Données d'entrée d'un 2^{e} exemple
```
7
3
0
6
```

English version appears before the French version

```
4
1
```

Données de sortie du 2^{e} exemple

```
T
```

Justification des données de sortie du 2^{e} exemple

Les Ananas ont marqué $7 \cdot 3 + 3 \cdot 2 + 0 \cdot 1 = 27$ points tandis que les Bananes ont marqué $6 \cdot 3 + 4 \cdot 2 + 1 \cdot 1 = 27$ points. Ils ont donc fait match nul.

English version appears before the French version

Problem J2: Time to Decompress

Problem Description
You and your friend have come up with a way to send messages back and forth.

Your friend can encode a message to you by writing down a positive integer N and a symbol. You can decode that message by writing out that symbol N times in a row on one line.

Given a message that your friend has encoded, decode it.

Input Specification
The first line of input contains L, the number of lines in the message.

The next L lines each contain one positive integer less than 80, followed by one space, followed by a (non-space) character.

Output Specification
The output should be L lines long. Each line should contain the decoding of the corresponding line of the input. Specifically, if line $i + 1$ of the input contained `N x`, then line i of the output should contain just the character `x` printed `N` times.

Sample Input
```
4
9 +
3 -
12 A
2 X
```

Output for Sample Input
```
+++++++++
---
AAAAAAAAAAAA
XX
```

La version française figure à la suite de la version anglaise.

Problème J2: Décompressons

Énoncé du problème
Votre ami et vous avez trouvé un moyen de vous envoyer des messages codés.

Votre ami peut encoder un message en écrivant un entier positif, N, et un symbole. Vous pouvez décoder ce message en écrivant ce symbole N fois dans une seule ligne.

Votre ami a encodé un message, décodez-le.

Précisions par rapport aux données d'entrée
La première ligne contiendra le nombre de lignes dans le message, soit L.

Les L lignes suivantes contiennent chacune un entier positif inférieur à 80, suivi d'un espace, suivi d'un caractère autre qu'un espace.

Précisions par rapport aux données de sortie
Il devrait y avoir L lignes dans les données de sortie. Chaque ligne des données de sortie doit contenir le décodage de la ligne correspondante des données d'entrée. Plus précisément, si la ligne $i + 1$ des données d'entrée contient `N x`, donc la ligne i des données de sortie devrait contenir uniquement le caractère `x` répété `N` fois.

Exemple de données d'entrée
```
4
9 +
3 -
12 A
2 X
```

Exemple de données de sortie
```
+++++++++
---
AAAAAAAAAAAA
XX
```

English version appears before the French version

Problem J3: Cold Compress

Problem Description
Your new cellphone plan charges you for every character you send from your phone. Since you tend to send sequences of symbols in your messages, you have come up with the following compression technique: for each symbol, write down the number of times it appears consecutively, followed by the symbol itself. This compression technique is called *run-length encoding*.

More formally, a block is a substring of identical symbols that is as long as possible. A block will be represented in compressed form as the length of the block followed by the symbol in that block. The encoding of a string is the representation of each block in the string in the order in which they appear in the string.

Given a sequence of characters, write a program to encode them in this format.

Input Specification
The first line of input contains the number N, which is the number of lines that follow. The next N lines will contain at least one and at most 80 characters, none of which are spaces.

Output Specification
Output will be N lines. Line i of the output will be the encoding of the line $i + 1$ of the input. The encoding of a line will be a sequence of pairs, separated by a space, where each pair is an integer (representing the number of times the character appears consecutively) followed by a space, followed by the character.

Sample Input
```
4
++++===!!!!
777777......TTTTTTTTTTT
(AABBC)
3.1415555
```

Output for Sample Input
```
3 + 3 = 4 !
6 7 6 . 12 T
1 ( 2 A 2 B 1 C 1 )
1 3 1 . 1 1 1 4 1 1 4 5
```

Explanation of Output for Sample Input
To see how the first message (on the second line of input) is encoded, notice that there are 3 + symbols, followed by 3 = symbols, followed by 4 ! symbols.

La version française figure à la suite de la version anglaise.

Problème J3: Compresse froide

Énoncé du problème
Votre nouveau forfait cellulaire vous coûte pour chaque caractère que vous envoyez. Comme vous avez tendance à envoyer des séquences de symboles dans vos messages, vous avez mis au point la technique de compression suivante pour chaque symbole: vous écrivez le nombre de fois où il paraît de manière consécutive et vous écrivez ensuite le symbole lui-même. Cette technique de compression s'appelle le *codage par plages*.

Dans un contexte plus formel, un bloc est une sous-chaîne de symboles identiques d'une grande longueur. On peut représenter un bloc sous sa forme comprimée en écrivant la longueur du bloc suivie par le symbole qui paraît dans le bloc. Une chaîne peut être encodée par la représentation de chaque bloc de cette chaîne dans l'ordre dans lequel ils y figurent.

Écrivez un programme qui encoderait une séquence de caractères dans ce format.

Précisions par rapport aux données d'entrée
La première ligne des données d'entrée contient le nombre de lignes qui suivront, soit N. Les N lignes suivantes contiennent de un à 80 caractères dont aucun n'est un espace.

Précisions par rapport aux données de sortie
Il devrait y avoir N lignes dans les données de sortie. La ligne i des données de sortie devrait contenir l'encodage de la ligne $i + 1$ des données d'entrée. L'encodage d'une ligne sera formé d'une séquence de couples qui seront séparés les uns des autres par des espaces. Chaque couple comportera un entier (qui représente le nombre de fois que le caractère paraît consécutivement), suivi d'un espace, suivi du caractère lui même.

Exemple de données d'entrée
```
4
++++===!!!!
777777......TTTTTTTTTTTT
(AABBC)
3.1415555
```

Exemple de données de sortie
```
3 + 3 = 4 !
6 7 6 . 12 T
1 ( 2 A 2 B 1 C 1 )
1 3 1 . 1 1 1 4 1 1 4 5
```

Justification des données de sortie
On examine la manière dont le premier message (qui se trouve sur la deuxième ligne des données d'entrée) a été encodé: on remarque qu'il y a le symbole + 3 fois, suivi du symbole = 3 fois, suivi du symbole `!` 4 fois.

English version appears before the French version

Problem J4/S1: Flipper

Problem Description
You are trying to pass the time while at the optometrist. You notice there is a grid of four numbers:

1	2
3	4

You see lots of mirrors and lenses at the optometrist, and wonder how flipping the grid horizontally or vertically would change the grid.

Specifically, a "horizontal" flip (across the horizontal centre line) would take the original grid of four numbers and result in:

3	4
1	2

A "vertical" flip (across the vertical centre line) would take the original grid of four numbers and result in:

2	1
4	3

Your task is to determine the final orientation of the numbers in the grid after a sequence of horizontal and vertical flips.

Input Specification
The input consists of one line, composed of a sequence of at least one and at most 1 000 000 characters. Each character is either `H`, representing a horizontal flip, or `V`, representing a vertical flip.

For 8 of the 15 available marks, there will be at most 1 000 characters in the input.

Output Specification
Output the final orientation of the four numbers. Specifically, each of the two lines of output will contain two integers, separated by one space.

Sample Input 1
```
HV
```

Output for Sample Input 1
```
4 3
2 1
```

La version française figure à la suite de la version anglaise.

Sample Input 2

```
VVHH
```

Output for Sample Input 2

```
1 2
3 4
```

La version française figure à la suite de la version anglaise.

Problème J4/S1: Inverser

Énoncé du problème

Vous essayez de passer le temps chez l'optométriste. Vous remarquez une grille de quatre nombres:

1	2
3	4

Vous remarquez d'ailleurs plusieurs miroirs et lentilles chez l'optométriste et vous vous demandez l'effet qu'aurait une inversion horizontale ou verticale sur la grille.

Plus précisément, une inversion horizontale (sur la ligne horizontale du milieu) de la grille originale donnerait la grille suivante comme résultat:

3	4
1	2

Tandis qu'une inversion verticale (sur la ligne verticale du milieu) de la grille originale donnerait la grille suivante comme résultat:

2	1
4	3

Votre tâche consiste à déterminer l'orientation finale des nombres dans la grille après une séquence d'inversions verticales et horizontales.

Précisions par rapport aux données d'entrée

Les données d'entrée ne contiennent qu'une seule ligne. Cette ligne est composée d'une séquence de caractères (dont au minimum un seul caractère et au maximum 1 000 000 caractères). Chaque caractère est soit un `H` (ce qui représente une inversion horizontale), soit un `V` (ce qui représente une inversion verticale).

Pour 8 des 15 points disponibles, il y aura au plus 1 000 caractères dans les données d'entrée.

Précisions par rapport aux données de sortie

Les données de sortie devraient afficher l'orientation finale des quatre nombres. Plus précisément, chacune des deux lignes de sortie contiendra deux entiers qui seront séparés par un espace.

Données d'entrée d'un 1er exemple

```
HV
```

Données de sortie du 1er exemple

```
4 3
2 1
```

English version appears before the French version

Données d'entrée d'un 2^{e} exemple

```
VVHH
```

Données de sortie du 2^{e} exemple

```
1 2
3 4
```

English version appears before the French version

Problem J5: Rule of Three

Problem Description
A *substitution rule* describes how to take a sequence of symbols and convert it into a different sequence of symbols. For example, ABA $\rightarrow$ BBB, is a substitution rule which means that ABA can be replaced with BBB. Using this rule, the sequence A**ABA**A would be transformed into the sequence ABBBA (the substituted symbols are in **bold**).

In this task, you will be given three substitution rules, a starting sequence of symbols and a final sequence of symbols. You are to use the substitution rules to convert the starting sequence into the final sequence, using a specified number of substitutions.

For example, if the three substitution rules were:

1. AA $\rightarrow$ AB
2. AB $\rightarrow$ BB
3. B $\rightarrow$ AA

we could convert the sequence AB into AAAB in 4 steps, by the following substitutions:

$$\mathbf{AB} \rightarrow \mathbf{B}\text{B} \rightarrow \text{AA}\mathbf{B} \rightarrow \text{AA}\mathbf{AA} \rightarrow \text{AAAB},$$

where the symbols to be replaced are shown in **bold**. More specifically, from the initial sequence AB, substitute rule 2 starting at position 1, to get the result BB. From BB, substitute rule 3, starting at position 1, to get the result AAB. From AAB, substitute rule 3, starting at position 3, to get the result AAAA. From AAAA, substitute rule 1, starting at position 3, to get the result AAAB, which is the final sequence.

Input Specification
The first three lines will contain the substitution rules. Each substitution rule will be a sequence of `A`'s and `B`'s, followed by a space following by another sequence of `A`'s and `B`'s. Both sequences will have between one and five symbols.

The next line contains three space separated values, S, I and F. The value S ($1 \leq S \leq 15$) is an integer specifying the number of steps that must be used, and the values I (the initial sequence) and F (the final sequence) are sequences of `A`'s and `B`'s, where there are at least one and at most 5 symbols in I and at least one and at most 50 symbols in F.

For 7 of the 15 marks available, $S \leq 6$.

For an additional 7 of the 15 available marks, $S \leq 12$.

Output Specification
The output will be S lines long and describes the substitutions in order.

La version française figure à la suite de la version anglaise.

Line i of the output will contain three space-separated values, R_i, P_i, and W_i:

- R_i is the substitution rule number (either `1`, `2` or `3`) that will be used.
- P_i is the starting position index of where the substitution rule will be applied in the sequence. Notice that the string is 1-indexed (i.e., the first character of the string is at index 1).
- W_i is the sequence that results from this substitution. Specifically, W_i is the sequence of symbols that results by applying substitution rule R_i starting at position P_i from the previous sequence of symbols, W_{i-1}, where we define W_0 to be the initial sequence I. Note that $W_S = F$, the final sequence.

There will always be at least one sequence of S substitutions that will convert I into F. If there is more than one possible sequence of substitutions, any valid sequence will be accepted.

Sample Input

```
AA AB
AB BB
B AA
4 AB AAAB
```

Possible Output for Sample Input

```
2 1 BB
3 1 AAB
3 3 AAAA
1 3 AAAB
```

Explanation of Output for Sample Input

This is the example outlined in the problem description. Note that the following is another possible valid substitution sequence:

```
2 1 BB
3 2 BAA
1 2 BAB
3 1 AAAB
```

Specifically, showing the substitutions in **bold**, we get

$$\mathbf{AB} \rightarrow \mathrm{B}\mathbf{B} \rightarrow \mathrm{B}\mathbf{AA} \rightarrow \mathbf{B}\mathrm{AB} \rightarrow \mathrm{AAAB}.$$

La version française figure à la suite de la version anglaise.

Problème J5: La règle de trois

Énoncé du problème
Une *règle de substitution* décrit la manière dont on peut convertir une séquence de symboles en une autre séquence de symboles. Par exemple, ABA $\rightarrow$ BBB, est une règle de substitution qui dit que la séquence ABA peut être remplacée par la séquence BBB. À l'aide de cette règle, on peut transformer la séquence A**ABA**A en la séquence ABBBA (les symboles qui ont subi une substitution sont en **gras**).

Dans cette tâche, vous recevrez trois règles de substitution, une séquence initiale de symboles, ainsi qu'une séquence finale de symboles. Vous devez utiliser les règles de substitution afin d'obtenir la séquence finale de symboles à partir de la séquence initiale en utilisant un nombre spécifié de substitutions.

Par exemple, si les trois règles de substitution étaient:

1. AA $\rightarrow$ AB
2. AB $\rightarrow$ BB
3. B $\rightarrow$ AA

On convertirait la séquence AB en la séquence AAAB en 4 étapes en effectuant les substitutions suivantes:

$$\mathbf{AB} \rightarrow \mathbf{B}\text{B} \rightarrow \text{AA}\mathbf{B} \rightarrow \text{AA}\mathbf{A}\mathbf{A} \rightarrow \text{AAAB},$$

où les symboles à remplacer sont en **gras**. Plus précisément, en commençant par la séquence initiale AB, on substitue la règle 2 dans la première position de cette séquence afin d'obtenir BB. On substitue la règle 3 dans la première position de la séquence BB afin d'obtenir AAB. On substitue la règle 3 dans la troisième position de la séquence AAB afin d'obtenir AAAA. On substitue la règle 1 dans la troisième position de la séquence AAAA afin d'obtenir la séquence finale de AAAB.

Précisions par rapport aux données d'entrée
Les trois premières lignes contiendront les règles de substitution. Chaque règle de substitution sera composée d'une séquence de `A` et de `B`, suivie d'un espace, suivie d'une autre séquence de `A` et de `B`. Les deux séquences auront chacune de un à cinq symboles.

La prochaine ligne contiendra les trois valeurs S, I et F dont chacune sera séparée des autres par un espace. La valeur S ($1 \leq S \leq 15$) est un entier qui spécifie le nombre d'étapes que l'on doit utiliser. La valeur I (la séquence initiale de symboles) et la valeur F (la séquence finale de symboles) sont des séquences de `A` et de `B`. La valeur I peut contenir de un à cinq symboles tandis que la valeur F peut contenir de un à cinquante symboles.

English version appears before the French version

Pour 7 des 15 points disponibles, $S \leq 6$.

Pour 7 autres points parmi les 15 points disponibles, $S \leq 12$.

Précisions par rapport aux données de sortie
Il devrait y avoir S lignes dans les données de sortie. Ces dernières décrivent les substitutions en ordre.

La ligne i des données de sortie contiendra les trois valeurs R_i, P_i et W_i dont chacune sera séparée des autres par un espace:

- R_i est le nombre de la règle de substitution qui sera utilisée (soit `1`, soit `2`, soit `3`).
- P_i est l'indice de position de départ de l'endroit où la règle de substitution sera appliquée dans la séquence. On remarque que le premier caractère de la chaîne a un indice de 1.
- W_i est la séquence qui est le résultat de cette substitution. Plus précisément, W_i est la séquence de symboles qui est produite après l'application de la règle de substitution R_i dans la position P_i de la séquence de symboles précédente, W_{i-1}, où W_0 est la séquence initiale I. On remarque que $W_S = F$, soit la séquence finale.

Il y aura toujours au moins une séquence de S substitutions qui convertit I en F. S'il y a plus d'une séquence possible de substitutions, toute séquence valide sera acceptée.

Exemple de données d'entrée
```
AA AB
AB BB
B AA
4 AB AAAB
```

Exemple de données de sortie possibles
```
2 1 BB
3 1 AAB
3 3 AAAA
1 3 AAAB
```

Justification des données de sortie
Ceci est l'exemple fourni dans l'énoncé du problème. Voici une autre séquence valide de substitutions:

```
2 1 BB
3 2 BAA
1 2 BAB
3 1 AAAB
```

English version appears before the French version

En indiquant les substitutions en **gras**, on obtient:

$$\mathbf{AB} \to \mathrm{B}\mathbf{B} \to \mathrm{B}\mathbf{AA} \to \mathbf{B}\mathrm{AB} \to \mathrm{AAAB}.$$

English version appears before the French version