2018 LGE Code Jam: Online Round 2

Problem A. Alex and Playing Cards

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

You have a deck of playing cards (that people use to play poker, for instance). A deck of playing cards consists of exactly 52 cards where each card has a suit (spades(♠), clubs(♣), diamonds(♢), or hearts(♡)) and a face value (for this problem, an integer value between 1 and 13, inclusive). For each one of the four suits and each one of the thirteen values, there exists exactly one card in a standard deck (hence 4×13 = 52 cards).

You are obsessed with hands of 'full house' in poker. A hand of full house consists of exactly five cards such that three of them share the same face value (call it $v1$) and the other two cards share the same face value (call it $v2$) where $v1 \neq v2$.

For instance, the following five cards make a full house hand.

- (♠, 1), (♣, 1), (♢, 1), (♠, 2), (♣, 2)

On the other hand, the following five cards do not.

- (♠, 1), (♡, 1), (♠, 2), (♣, 2), (♢, 3)

Using a standard deck of cards, you can make 8 hands of full house simultaneously (when each card can be used for at most one hand). There are many ways to do so, but here is one example:

- (♠, 1), (♣, 1), (♢, 1), (♠, 13), (♣, 13)
- (♠, 2), (♣, 2), (♢, 2), (♠, 12), (♣, 12)
- (♠, 3), (♣, 3), (♢, 3), (♠, 11), (♣, 11)
- (♠, 4), (♣, 4), (♢, 4), (♠, 10), (♣, 10)
- (♠, 5), (♣, 5), (♢, 5), (♡, 13), (♢, 13)
- (♠, 6), (♣, 6), (♢, 6), (♡, 12), (♢, 12)
- (♠, 7), (♣, 7), (♢, 7), (♡, 11), (♢, 11)
- (♠, 8), (♣, 8), (♢, 8), (♡, 10), (♢, 10)

Alex, a friend of yours, recently borrowed your standard deck of cards (52 cards), but he says he has lost a few cards. He hasn't told you which cards he lost. You are curious how many hands of full house you can make with the remaining cards simultaneously, if you try your best. That is, using the remaining cards, you will always try to maximize the number of hands of full house that you can make simultaneously. Naturally, depending on which cards Alex has lost, the number of hands may vary.

For instance, suppose that Alex has lost 42 cards altogether, and only 10 cards are remaining. If the 10 cards remaining happen to be the following, then you can still make 2 hands of full house at the same time.

- (♠, 1), (♢, 1), (♣, 2), (♠, 2), (♢, 2)
- (♠, 11), (♢, 11), (♣, 12), (♠, 12), (♢, 12)

On the other hand, if the remaining 10 cards are all clubs (no other suits), then you cannot make any hand of full house. Notice that, in this case, the minimum and maximum number of hands of full house you can make (when you try your best) are 0 and 2, respectively.

Only knowing how many cards Alex has lost, you want to know the minimum and maximum value of the maximum number of hands of full house you can make simultanouesly if you try your best.

Input

The first line describes the number of missing cards, n, which is between 0 and 52, inclusive.

Output

You must output a single line that contains two numbers, describing the minimum and maximum value of the maximum number of hands of full house you can make simultaneously.

Subtask 1 (4 points)

- $0 \leq n \leq 52$

Examples

standard input	standard output
42	0 2

This was described in the problem statement.

standard input	standard output
47	0 1

This is similar to the first sample – it depends on whether the remaining five cards can make a full house hand or not.

standard input	standard output
0	8 8

Alex did not loes any card. We can make 8 hands of full house as described in the problem statement.

standard input	standard output
4	8 8
20	3 6

Problem B. Operations on Array of Integers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

You are going to apply a few operations on an array of integers X.

The array X contains n integers, which are initialized to 0's at the beginning. You are going to apply exactly q operations to X.

There are precisely two kinds of operations: addition and reversal. When you apply the addition operation to X, you are going to add value i to $X[i]$ (so $X[i]$ becomes its old value plus i) where i is between 0 and $n-1$, inclusive (we use 0-baed index). When you apply the reversal operation to X, you are also given two indices $(i,\ j)$ where $i \leq j$ such that you are going to reverse the order of elements $X[i], X[i+1], \ldots, X[j-1], X[j]$.

For instance, suppose that $n = 6$, and you apply the following five operations in order:

1. addition
2. addition
3. reversal $i = 0, j = 4$
4. addition
5. reversal $i = 2, j = 5$

In this example, X begins as [0, 0, 0, 0, 0, 0].

1. After the first operation, X becomes [0, 1, 2, 3, 4, 5].
2. After the second operation, X becomes [0, 2, 4, 6, 8, 10].
3. After the third operation, X becomes [8, 6, 4, 2, 0, 10].
4. After the fourth operation, X becomes [8, 7, 6, 5, 4, 15].
5. After the fifth (last) operation, X becomes [8, 7, 15, 4, 5, 6].

You are wondering about the values of certain elements of X, after you apply all of the q operations. Specifically, you are given an integer m and an array of indices (containing m elements) – let's call this array of indices Y. You want to know $X[Y[i]]$ for all i between 0 and $m-1$, after you finish applying the operations.

Input

The first line contains three integers n, q, and m ($1dnd1,000,000,0dqd10,000,1dmd5,000$). The following q lines describe operations, one operation per line. In each line, the line contains either '`a`' (one character) or '`r`', i, and j where i and j are indices of X (integers between 0 and $n-1$, inclusive) and $i \leq j$. Then the last line contains m integers (the array Y) that are indices of X (these indices are between 0 and n-1, inclusive).

Output

You must output exactly m lines where each line contains the value $X[Y[i]]$ after you apply the q operations.

Subtask 1 (5 points)

- $1 \leq n, q, m \leq 200$

Subtask 2 (6 points)

- $1 \leq n \leq 1,000,000$
- $0 \leq q \leq 10,000$
- $1 \leq m \leq 5,000$

Examples

standard input	standard output
6 5 6 a a r 0 4 a r 2 5 0 1 2 3 4 5	8 7 15 4 5 6

This example was discussed in the problem statement.

standard input	standard output
3 0 3 2 1 0	0 0 0
7 6 7 a a r 0 3 a a r 3 5 0 1 2 3 4 5 6	6 6 6 20 16 6 24
2 7 2 a r 0 1 a a r 0 1 a r 0 1 0 1	2 2

standard input	standard output
2 10 2 a r 0 1 a a r 0 1 a r 0 1 r 0 0 r 1 1 r 0 1 0 1	2 2

Problem C. Matrix Multiplication

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

You are given n matrices, labeled as $M_1, M_2, \ldots, M_n$. Matrix M_i (where $i = 1, 2, \ldots, n$) has $R[i]$ rows and $C[i]$ columns.

For each i between 1 and n (inclusive), you want to compute a number (we call V_i) as follows.

Using the matrices $M_1, M_2, \ldots, M_i$, you will first determine if you can find an ordering of these i matrices such that you can multiply all i of them. Note that two matrices of size $R \times C$ and $R' \times C'$ can be multiplied if and only if $C = R'$, and the resulting matrix is of size $R \times C'$. If this is impossible, $V_i = 0$. If this is possible, then V_i is defined as the size of the largest matrix you can obtain in this manner. In other words, among all orderings of the i matrices, V_i is the maximum number of elements found in the matrix that is the product of the i matrices.

You are asked to compute the values $V_1, V_2, \ldots, V_n$.

Input

The first line contains a positive integer n ($1 \leq n \leq 1,000$). Each of the following n lines contains two numbers each, denoting the number of rows and the number of columns of a matrix. The number of rows and the number of columns of each matrix is between 1 and 1,000, inclusive.

Output

You must output n lines where the i-th line contains the value V_i described in the problem statement.

Subtask 1 (5 points)

- $1 \leq n \leq 10$
- $1 \leq R, C \leq 1,000$

Subtask 2 (10 points)

- $1 \leq n \leq 1,000$
- $1 \leq R, C \leq 1,000$

Examples

standard input	standard output
3 8 2 2 8 2 2	16 64 64

When $i = 1$, the answer is trivially 16 as M_1 contains 16 elements.

When $i = 2$, we can multiply them ($M_1 \times M_2$) to obtain a 8 x 8 matrix.

Note that we can also multiply them in the other order ($M_2 \times M_1$) to obtain a 2 x 2 matrix, but it is not maximum.

When $i = 3$, we can multiply the three matrices as $M_1 \times M_3 \times M_2$, to obtain a 8 x 8 matrix.

standard input	standard output
3 4 9 9 1 9 9	36 4 4

standard input	standard output
3 4 10 10 1 10 6	40 4 0

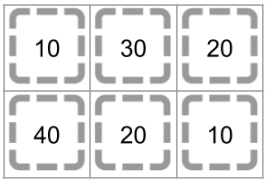
There is no ordering that results in a valid matrix multiplication of the three matrices.

standard input	standard output
3 10 3 3 10 5 5	30 100 0
4 1 1 1 1 1 1 1 1	1 1 1 1

Problem D. City Planning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

You have been appointed as the director of city planning, and your main role is to determine where skyscrapers to be built and where parks will be built. The city consists of $R \times C$ city blocks such that blocks are laid out in R rows and C columns (you can consider it as a grid of size $R \times C$). Different city blocks are different distances above the sea level, which makes constructions difficult. Each city block is currently empty, and the goal is to build something on every block – whether it is a skyscraper or a park. For instance, the following image shows a city grid that consists of two rows and three columns (totaling six city blocks). Each city block has an associated integer value with it, which tell how many meters the block is above the sea level.



The city council asked you to find a way to build skyscrapers and parks, while meeting the following rules.

1. A skyscraper must be built on a single city block (it can occupy only one city block).
2. Each city block must either have a skyscraper built on it OR have no skyscraper but a part of a park built on it.
3. If a skyscraper is built on a city block of level y, then all city blocks that are at most y meters above the sea level must also have a skyscraper (on each of them).
4. Each park must be built on two adjacent city blocks, and no two parks can overlap (share the same city block). In particular, no park can be built out of the city grid (the $R \times C$ grid).

Because of the third rule, you must first choose some non-negative integer Z such that all city blocks that are at most Z meters above the sea level will have skyscrapers built on – this makes the construction much easier. The remaining city blocks (that do not have skyscrapers on them) will have parks built. The citizens do not like small parks, and therefore each park requires two adjacent blocks in order to be properly constructed (this is the fourth rule). Two city blocks are adjacent if they share a road (or an edge) between them, but not diagonally (hence each city block has at most four adjacent city blocks).

The city council asks you to compute the minimum cost of building skyscrapers and parks while meeting the above conditions. Each park (occupying two adjacent city blocks) costs D dollars. For skyscrapers, for your choice of Z (non-negative integer), the total cost will be $Z \times W$ dollars where W is a positive integer constant. Recall that you get to choose the value of Z, and all city blocks that are at most Z meters above the sea level (whose Y-values are at most Z) will have skyscrapers on them.

Given these requirements, compute the minimum cost of building skyscrapers and parks such that no city block is left empty.

Input

The first line will contain four integers R, C, D, and W where $1 \leq R, C \leq 50$ and $1 \leq D, W \leq 10^9$. Each of the following R lines will contain C numbers each, decribing the y values of the city blocks. The value of each y is between 1 and 10^9, inclusive.

Output

You must output the minimum cost of buildling skyscrapers and parks.

Subtask 1 (6 points)

- $1 \leq R, C \leq 5$
- $1 \leq D, W, y \leq 100$

Subtask 2 (11 points)

- $1 \leq R, C \leq 20$
- $1 \leq D, W, y \leq 1,000,000$

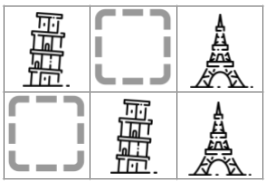
Subtask 3 (13 points)

- $1 \leq R, C \leq 50$
- $1 \leq D, W, y \leq 1,000,000,000$

Examples

standard input	standard output
2 3 120 10 10 30 20 40 20 10	340

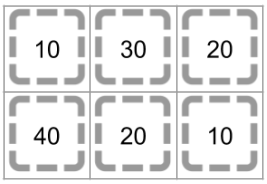
In this case, choosing $Z = 10$ leads to building two skyscrapers (at 10*10 = 100 dollars) and constructing two parks to cover the remaining four city blocks (at 120*2 = 240 dollars), totalling 340 dollars. This is optimal (there are many values of Z that achieces this). On the other hand, if you choose Z = 0 then the total cost is 360 dollars; if you choose $Z = 40$, the total cost is 400 dollars. Note that if you choose $Z = 20$, you will be building four skyscrapers but you will not be able to build a park on the remaining two blocks in a legitimate way, as the following image demonstrates.



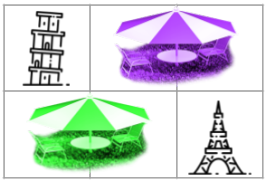
standard input	standard output
1 2 76 59 62 77	76
1 4 71 87 41 2 95 68	142
3 3 16 12 58 90 8 81 38 94 12 56 80	160
4 4 71 3 85 86 97 99 30 74 50 88 77 78 34 83 59 26 54 63	297

Notes

For Python users, using PyPy is recommended.

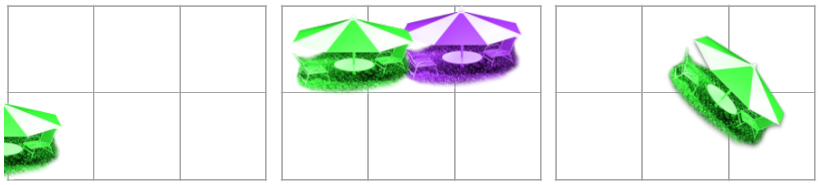


In the same example as above, if you choose $Z = 10$, for instance, two skyscrapers will be built on two city blocks (on the top-left and bottom-right city blocks), and two parks can be built on the remaining four city blocks. This is demonstrated in the following image.



The following three examples show what is NOT allowed when it comes to building a park.

In the left example, a park is partially built on the bottom-left city block, which is not allowed as it goes beyond the city grid. In the middle example, two parks are overlapping on the same city block (the top-middle block). In the right example, a park is not built on two adjacent city blocks. All these examples show wrong ways of building parks, which will not be approved by the city council.



Problem E. New Language

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

You have designed a new language. In this language, a word is defined as a non-empty string that meets the following conditions.

1. It is a non-empty string that only contains lower-case English alphabets (‘a’-‘z’) and some of special characters – periods (‘.’), hyphens (‘-’), and asterisks (‘*’).
2. No three consecutive characters are all special characters (special characters are ‘.’, ‘-’, and ‘*’).
3. No three consecutive vowels are found (there are five vowels: ‘a’,‘e’, ‘i’, ‘o’, ‘u’).
4. No three consecutive characters are of the same character (regardless of them being consonants, vowels, and specials).
5. Neither the first character nor the last character is a special character.

For instance, there are 26 valid words of length 1 in this language – each lower-caes English alphabet itself is a valid word. In addition, “boj”, “i.am.coder”, and ”hello*world” (quotes for clarity) are valid words, whereas “zzz”, “.boj”, “mioum”, and “hello-.-world” are not valid words.

Given two positive integers a, and b, compute the number of valid words of lengths between a and b (inclusive). Since this number can be quite large, you should output this number modulo $(10^9 + 7)$.

Input

The first line will contain two integers a and b ($1 \leq a \leq b \leq 10^{18}$).

Output

The first line must contain the number of valid words of lengths between a and b (inclusive) modulo $(10^9 + 7)$.

Subtask 1 (5 points)

- $1 \leq a \leq b \leq 5$

Subtask 2 (6 points)

- $1 \leq a \leq b \leq 1,000$

Subtask 3 (9 points)

- $1 \leq a \leq b \leq 1,000,000$

Subtask 4 (20 points)

- $1 \leq a \leq b \leq 10^{18}$

Examples

standard input	standard output
1 1	26
1 2	702
4 5	16745010
28 28	9496760
999999999999999999 999999999999999999	415475129