培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE

6TH · 2022

AUG 13, 2022 (SATURDAY)

LA SALLE COLLEGE

ID	NAME	TIME LIMIT	MEMORY LIMIT
A	Advertere Augmento	0.5 seconds	256 MB
B	Balanced Splitting	0.5 seconds	256 MB
C	Carpark	0.5 seconds	256 MB
D	Delivery	0.5 seconds	256 MB
E	Exclusive-or Merging	0.5 seconds	256 MB
F	Fall Guys	0.5 seconds	256 MB
G	Great Plummet	0.5 seconds	256 MB
H	HDL Shipping Service	4 seconds	256 MB
I	I want to buy games!	0.5 seconds	256 MB
J	Just Another FFT Problem	0.5 seconds	256 MB
K	Karaoke	0.5 seconds	256 MB
L	Linear Game	0.5 seconds	256 MB

Pascal	`ppcx64-3.0.0 -O2 -Sg -v0 -dEVAL -XS`
C	`gcc-7 -static -Wno-unused-result -DEVAL -lm -s -O2`
C++17	`g++-7 -static -std=c++17 -Wno-unused-result -DEVAL -lm -s -O2`

Compilation Time Limit: 10 seconds Memory Limit: 512 MB

AUTHORS / PREPARED BY

Cheng Hei Chit, Chiu Long Hin Vincent, Chung Wai Jit, Lee Cheuk Kit,
Lee Ching Hei, Lee Kai Fung Kevin, Ng Yau Fu, Wai Ka Hei,
Wong Man Hang, Xie Lingrui, Yeung Man Tsung, Yuen Lok Kan Ethen

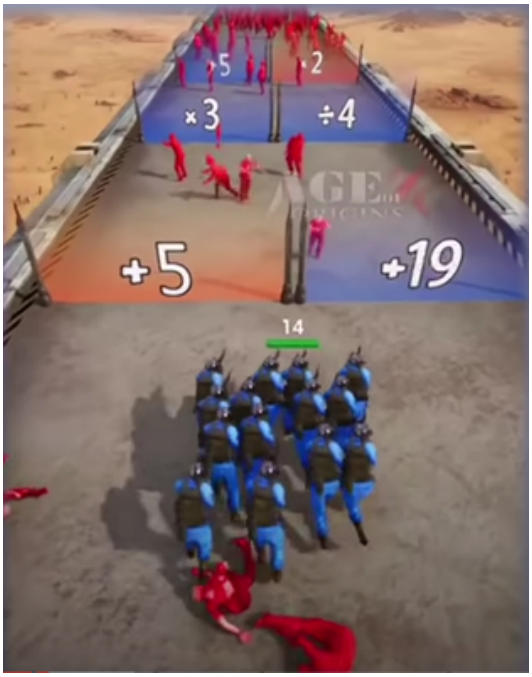
NOTES

1. Output strings are checked case-sensitively.
2. Trailing spaces in output lines are allowed.
3. The end-of-line character at the end of the output is optional.
4. Using scientific notation for real number output is not allowed.
5. 64-bit integer types (e.g. `int64` for Pascal, `long long` for C/C++) may be required for some problems. The C format string token is `%lld`.

A Advertere Augmento

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Recently, Alice has come across an interesting game advertisement when scrolling through social media. The game involves a squad of soldiers going through gates with arithmetic operations that would apply to their fighting power.



The game advertisement.

Once she downloaded the game, she realized that the advertisement was fake and the actual gameplay was nowhere like what was portrayed in the ads. "What a pity!" She thought, "It would make a good mathematical game." Hence she decides to recreate one on her own.

The game she created is as follows:

- The character starts with a **Power Level** of 0.
- There are N stages. Each stage has a pair of **Magic Gates**.
- The character needs to pass through the N stages in order by choosing 1 of the 2 gates to pass through at each stage.
- Each **Magic Gate** is associated with an arithmetic operation, for example, $+2$, $-(-3)$, $*7$. The operator must be one of add ($+$), subtract ($-$), or multiply ($*$), followed by an integer operand.
- Passing through the magic gate would apply the operation onto the character's current **Power Level**, for example, a character with **Power Level** 7 passing through a $+2$ magic gate would turn his **Power Level** into $7 + 2 = 9$.

It should be noted that different from the original game, the character is allowed to have a negative Power Level.

Given the configurations of all **Magic Gates**, could you figure out what is the maximum final **Power Level** that can be achieved after passing through all N gates when selecting optimally?

Input

The first line consists of an integer, N, the number of stages the character has to pass through.

The following N lines each denote the pair of Magic Gates at a stage, given in increasing distance from the starting position. The i^{th} of the N lines contains 4 space-separated variables, $C_{i,1}$, $V_{i,1}$, $C_{i,2}$ and $V_{i,2}$.

C_i is one of the following symbols `+` (addition), `-` (subtraction), `*` (multiplication), while V_i is an integer. $C_{i,1}$ $V_{i,1}$ and $C_{i,2}$ $V_{i,2}$ represent the arithmetic operations on the two gates respectively.

The input guarantees that the Power Level of the character would not exceed the 32-bit integer range at any point in time no matter which gate is selected.

$1 \leq N \leq 10^5$, $-10^9 \leq V_{i,1}, V_{i,2} \leq 10^9$

Output

Output a single integer, the maximum final **Power Level** that can be achieved.

Examples

input

```
3
+ 2 + 7
- 1 - 4
* 2 + 5
```

output

```
12
```

input

```
5
+ 1 + 1
* -1 * 2
+ -2 + 5
+ 5 - 3
* 1 * 3
```

output

```
36
```

input

```
2
- 1 - 1
+ 2 + 5
```

output

```
4
```

B Balanced Splitting

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

The author of this task dislikes long stories in task statements, so he is making the task statement of this task as concise as possible!

Given a 01-string S of length N, you have to process Q queries on it.

For the i^{th} query, you are given L_i and R_i, and you have to find any pair (a, b) such that $L_i \leq a \leq b \leq R_i$ and $S[a..b]$ contains exactly half of the number of `0`s and half of the number of `1`s in $S[L_i..R_i]$. If no such pair exist, output `-1`.

Input

The input consists of $Q + 2$ lines.

The first line of the input consists of 2 integers N, Q ($1 \leq N, Q \leq 10^5$), the length of the 01-string, and the number of queries respectively.

The second line of the input consists of the 01-string S of length N.

The i^{th} of the remaining Q lines each consist of two integers L_i, R_i ($1 \leq L_i \leq R_i \leq N$), the range of the query.

Output

Output Q lines, one for each query.

If a pair (a, b) exists, output a and b on the same line, separated by a space. If there are multiple solutions, output any.

If no such pair exists, output `-1` on the line.

Example

input

```
9 5
000101101
2 3
3 6
2 9
1 4
1 6
```

output

```
3 3
4 5
5 8
-1
3 5
```

Note

$S[x..y]$ denotes the substring of S from the x-th position to the y-th position inclusively. That is, $S[x..y] = S_x S_{x+1} S_{x+2} \ldots S_{y-1} S_y$.

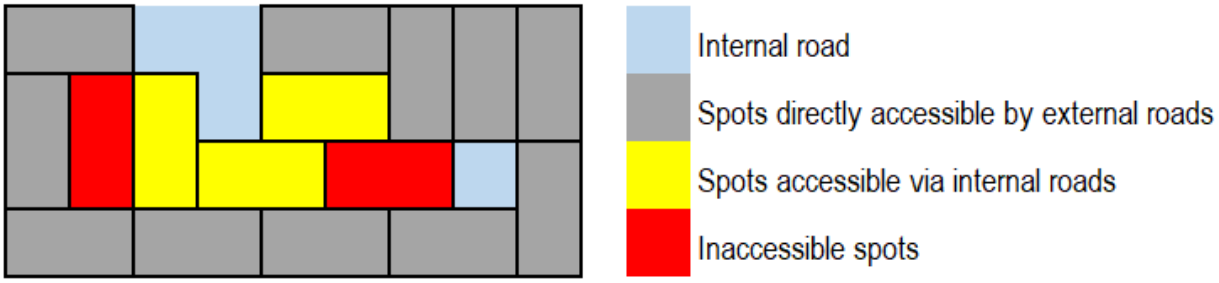
This page is intentionally left blank.

C Carpark

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Percy is a student studying at Heung Shing Secondary School. On the ground floor of the school, there is a piece of land for building a carpark. The piece of land can be represented by a grid with N rows and M columns. The land is surrounded by roads on all 4 sides. One day, the school principal asks Percy to redesign the carpark's layout.

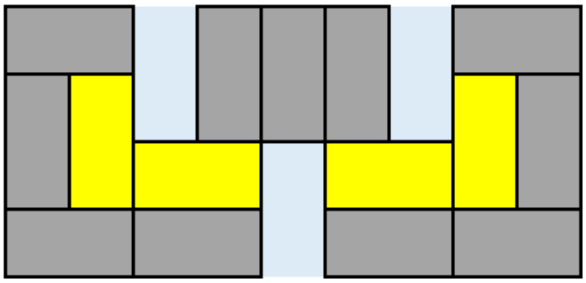
A parking spot requires 2 connected cells, i.e. a vertical or horizontal 1×2 region. Parking spots that touch the boundary of the land can be directly accessed by the external roads. Cells that are not part of a parking spot form the internal roads. A parking spot is also *accessible* if it is possible to reach the boundary of the land from the parking spot via some internal roads.



The principal has decided to give the Student of the Year award to Percy if he manages to design a layout that meets all of the following criteria:

- All parking spots are *accessible*.
- The total area of all parking spots is at least 80% of the land's area.

For example, in the layout below, the parking spots are all accessible by external and internal roads and the total area is $30/36 \approx 83.3\%$. Therefore it is an acceptable layout.



As a friend of Percy, can you help him to win the Student of the Year award?

Input

The input contains only one line with two integers, N and M ($2 \leq N, M \leq 100$).

Output

The output should contain N lines, each of which contains M integers separated by spaces, representing the layout of the carpark.

For the cells that are not part of a parking spot, you should use the integer 0 to represent them.

For each parking spot, you should use a **unique** positive integer between 1 and $2^{31}-1$ to represent it. So for each positive integer between 1 and $2^{31}-1$, it should only appear 0 or 2 times in the output.

It can be shown that a solution always exist with the given constraints.

Example

input

```
4 9
```

output

```
1 1 0 3 4 5 0 2 2
6 7 0 3 4 5 0 8 9
6 7 10 10 0 11 11 8 9
12 12 13 13 0 14 14 15 15
```

D Delivery

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

A long, long time ago, the Olympians resided in the idyllic Olympic village. The village was built in the form of a connected, undirected graph with N houses (vertices) and M roads (edges). Once every year, the Olympians gather in one of the houses and take part in a lavish feast.

Choosing which house to hold the feast in was not a straightforward task — instead of picking a house randomly, the Olympians were adamant that the *inconvenience* should be minimized. Since Olympians in every house were responsible for delivering food and gifts to the location of the feast, the *inconvenience* of holding the feast in house i was defined to be the maximum number of roads that an Olympian travels along to reach house i, assuming everyone walks to the feast using the smallest possible number of roads.

Unfortunately, we still do not know much about the structure of the graph to this day. Careful inspection of an ancient scroll revealed a note which stated the largest and smallest *inconvenience* among all N possible houses. Given these two numbers, can you reconstruct the layout of the Olympic village?

Input

The first and only line of input consists of two integers X and Y ($1 \leq X, Y \leq 100$), the largest and smallest *inconvenience* respectively.

Output

If there does not exist a graph with largest *inconvenience* X and smallest *inconvenience* Y, output `-1`.

Otherwise, output a possible graph with largest *inconvenience* X and smallest *inconvenience* Y. If there are multiple solutions, you may output any one of them.

On the first line, output two integers N ($2 \leq N \leq 1000$) and M. Then output M more lines, each containing two integers u and v ($1 \leq u, v \leq N$, $u \neq v$) denoting a road between house u and house v. Multiple edges between the same pair of houses are not allowed.

It is guaranteed that if there exists a solution, there exists a solution with $2 \leq N \leq 1000$.

Examples

input	output
4 2	7 6 3 5 6 2 2 1 3 4 2 3 7 5

input	output
8 13	-1

Note

For the first sample output, the table below shows the *inconvenience* of holding the feast in each house:

House	Inconvenience
1	4
2	3
3	2
4	2
5	3
6	4
7	4

E Exclusive-or Merging

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

"XOR Merging" is a game created by Ethen. In the game, the player is given two 01-strings S and T. The player can "xor merge" any two consecutive digits of S multiple times, until S becomes the target string T. When the player "xor merges" the two digits S_i and S_{i+1}, they will be replaced by the result of S_i xor S_{i+1}.

For example, let S be 100010111 and T be 101010, the player can convert S to T using three "xor merges".

1. First merge $S_2 = 0$ and $S_3 = 0$ into 0, so that S becomes 10010111.

2. Then merge $S_3 = 0$ and $S_4 = 1$ into 1, so that S becomes 1010111.

3. At last merge $S_6 = 1$ and $S_7 = 1$ into 0, so that S becomes 101010, which is the same as T.

However, Ethen found that it is not always possible to convert a given S to a given T by "xor merging". Can you help Ethen determine if a solution exists?

Input

The first line of the input contains the string S.

The second line of the input contains the string T.

Both S and T contains only `0`s and `1`s.

$1 \leq |T| \leq |S| \leq 10^6$, where $|S|$ is the length of string S and $|T|$ is the length of string T.

Output

Output `Yes` if it is possible to convert S to T by "xor merging", otherwise output `No`.

Examples

input

```
100010111
101010
```

output

```
Yes
```

input

```
11
1
```

output

```
No
```

This page is intentionally left blank.

F Fall Guys

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

In Fall Guys squads show, squads of 4 to compete against other teams. The show starts with 15 squads (60 players). In each round some squad(s) will be eliminated until one single squad remains which will be declared as the winner.

Now, your squad is competing in one of the rounds - Slime Climb. In Slime Climb, individual players race to the finish line as quick as possible while avoiding falling into the rising slime. Players who finish the race will score $(\text{number of players}) - (\text{individual rank}) + 1$ points. For example, when there are 60 players, the first player to reach the finish line will gain 60 points for their squad. The 2nd player will gain 59 points, etc. Other players who fall into the slime will score 0 points. Because the slime rises continuously, players who are slow will eventually fall into the slime. The total score of a squad is the sum of individual scores of the 4 members.

There are N squads ($4N$ players) including your squad. The squads are numbered from 1 to N. There is a target number of qualifying teams M. Normally, if the squads at the M^{th} and $(M+1)^{th}$ positions have different scores, the top M squads with the highest total score will qualify and the rest will be eliminated. If the squads at the M^{th} and $(M+1)^{th}$ positions have the same score, squads that tie for the M^{th} position will all be eliminated. However, when top $M+1$ squads all have the same score, the other squads that score less will be eliminated instead.

In the middle of the race, F players have reached the finish line. The i-th player to reach the finish line is from squad A_i. Also, S players fell into the slime. Their squad numbers are $B_1, B_2, \ldots, B_S$.

You would like to know if the set of qualifying teams have been determined already, regardless of the outcome (order of finishing the race or whether they fall into the slime) of the remaining players. To save time, the game can end the round early if the set has been determined.

Input

The first line contains 2 integer N and M. ($1 \leq M < N \leq 15$)

The second line begins with a non-negative integer F, followed by F integers $A_1, A_2, \ldots, A_F$.

The third line begins with a non-negative integer S, followed by S integers $B_1, B_2, \ldots, B_S$.

$0 \leq F + S < 4N$. It is guaranteed that a squad will not appear more than 4 times in the lists.

Output

Output `Yes` if the set of qualifying teams has been determined regardless of the outcome of the remaining players. Output `No` otherwise.

Examples

input

```
2 1
6 1 1 1 2 2 2
0
```

output

```
Yes
```

input

```
3 2
9 1 2 3 2 3 1 3 1 2
2 1 2
```

output

```
No
```

Note

In the first example, 3 players from the squad 1 already gained $8 + 7 + 6 = 21$ points and 3 players from squad 2 gained $5 + 4 + 3 = 12$ points. There is no way for squad 2 to gain more points to tie/surpass squad 1 and therefore the set of qualifying squads has been determined. Only squad 1 will qualify.

In the second example, all squads currently have 24 points. Only 1 player from squad 3 remains. If the player finishes the race, squad 3 will have 26 points and qualify. Squad 1 and 2 will be eliminated because they tie for the 2nd position. If the player fall into the slime, all teams will qualify because they have the same score.

Additional Notes:

- This is not exactly how the game works. In the animation above, the player is the only remaining player and the squad is already guaranteed 1st place but the round does not end early.
- In the actual game, some squads may not have 4 players and the number of players in a round may not be $4N$. For simplicity, in this problem we assume all squads have 4 players.

G Great Plummet

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

In the stock market, we like to analyse the change in price within some time intervals. Sometimes, it may provide insights for us to predict the price trends. In this task, you are required to analyse every ($M+1$)-day interval over the $N+1$ days of the market.

The N changes in prices between days are given. Let P_i be the stock price of Day i, A_i denoted the change $P_{i+1}-P_i$.

We are interested in finding the maximum continuous decline in the stock price at some time intervals. For each $1 \leq i \leq N$, we want to find that for the interval of last M values up to A_i, i.e. the range $max(1, i-M+1)$ to i (inclusive) in A.

For example, if we are considering the interval with 7 values $[-3,-5,2,-3,-2,-2,4]$. There are 2 ranges of continuous decline, which are $[-3,-5]$ and $[-3,-2,-2]$. The sum of the first range evaluates to -8, and the sum of the second range evaluates to -7. Since the magnitude of -8 is larger than -7, the maximum continuous decline of this interval is -8.

If the interval only contains non-negative numbers, the maximum continuous decline should be considered as 0.

Please find the answers to all intervals, $max(1, i-M+1)$ to i (inclusive) for all $1 \leq i \leq N$.

Input

The first line contain 2 integers, N and M.

The second line contains N integers, $A_1, A_2, \ldots, A_N$.

$1 \leq M \leq N \leq 10^5$.

$-10^9 \leq A_i \leq 10^9$.

Output

On the first line output N integers, the i-th integer representing the required answer to the interval $[max(1, i-M+1), i]$ of A.

Examples

input

```
10 4
4 -3 1 -2 -3 2 0 -1 -1 -2
```

output

```
0 -3 -3 -3 -5 -5 -5 -3 -2 -4
```

input

```
12 7
-3 -7 -1 -3 -5 6 -3 -2 -2 4 -1 2
```

output

```
-3 -10 -11 -14 -19 -19 -19 -16 -9 -8 -7 -7
```

This page is intentionally left blank.

H HDL Shipping Service

Time Limit: 4 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

HDL is a multinational corporation which is renowned in logistics shipping service. You, a computer science student, would love to enroll their technology graduate trainee program. After 5 rounds of interview, you have made it to the final one, which is a technical round with the senior manager. He gave you an intricate problem that the company is encountering at the time:

In the HDL logistics network, there are N shipping stations and M 1-way shipping routes, meaning that station U_i can send goods to V_i with D_i days. Among the N shipping stations, there are P gift's locations, Q receive locations and 1 headquarter. Each of the P gift's locations has A_i gifts in the inventory and they can send the gifts to other stations through the network. Each of the Q receive locations need to receive B_i gifts. For the headquarter, it has unlimited number of gifts in the inventory but all gifts sent from headquarter spend double days to pass through a shipping route (all D_i are doubled). Please note that one station can be headquarter, gift's location and receive location at the same time. Also you can send unlimited gifts through a shipping route at the same time. Find the earliest day that all Q receive locations receive at least B_i gifts, or tell it is impossible.

Struggling right after you heard the problem, the senior manager smirked, "Don't worry, many interviewees failed this task, you are just one of them...". To stand on your dignity, please solve the problem (and get the job).

Input

The first line consists of 4 integers, N, M, P and Q, which corresponds to the number of shipping stations, 1-way shipping routes, gift's locations and receive locations the network has.

Input is then followed by M lines, each having 3 integers, U_i, V_i and D_i, denoting that a 1-way shipping route exists between U_i and V_i, and gifts going through which would spend D_i days, and $2 \times D_i$ days for all gifts sent from the headquarter.

For the following P lines, each has 2 integers, X_i and A_i, meaning that there is a gift's location at station X_i with the inventory of A_i gifts. Similarly, for the next Q lines, each has 2 integers, Y_i and B_i, indicating that there is a receive location at station Y_i that needs to receive B_i gifts.

The last line of the input consists of 1 integer Z, meaning that the headquarter is located at station Z.

$1 \leq N \leq 10^5$, $0 \leq M \leq 2 \times 10^5$, $1 \leq P, Q \leq 50$

$1 \leq U_i, V_i, X_i, Y_i, Z \leq N$

$1 \leq A_i, B_i \leq 10^5$, $1 \leq D_i \leq 10^9$

All X_i are distinct. All Y_i are distinct.

Output

Output a single integer: the earliest day that all Q receive locations receive at least B_i gifts. If it is impossible, output -1 instead.

Examples

input

```
3 3 1 3
1 2 4
2 3 7
1 3 11
2 1
1 3
2 1
3 1
1
```

output

```
8
```

input

```
2 1 1 1
1 2 11
2 100
1 1
2
```

output

```
-1
```

Note

In the first sample, the optimal solution will be:

- The gift's location at station 2 shall send 1 gift to the receive location at station 3. It will take 7 days.
- The headquarter shall send gifts to all other receive locations. It will take $2 \times 4 = 8$ days to reach station 2.

I I want to buy games!

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Ian recently bought a Nintendo Switch using the consumption voucher. He wanted to buy games from Nintendo e-shop but found that the games are way too expensive. Fortunately, the games are discounted on e-shop from time to time.

Ian sees that there are N games listed on e-shop, with a price of $A_1, A_2, \ldots, A_{N-1}, A_N$ dollars respectively. He has D days (Day 1 to Day D) to buy games with a total budget of K dollars. Since Ian needs time to play the game, he would buy a maximum of 1 game each day (can be 0). Also, he would not buy the same game more than once.

Noted that counting from Day 1 (today), the i-th game would be put to a discounted price of P_i (which is obviously smaller than A_i) if that day is a multiple of i. For example, the 3-rd game would be sold at price P_i at day 3, 6, 9, ... and so on.

Please help Ian to find the maximum number of games he could buy within his budget.

Input

On the first line, there are 3 space-separated integers N, K and D, the number of games listed, Ian's budget in dollars and the number of days for Ian to buy games.

On the second line, there are N integers, $A_1, A_2, \ldots, A_{N-1}, A_N$, the price of each game.

On the third line, there are N integers, $P_1, P_2, \ldots, P_{N-1}, P_N$, the discounted price of each game.

$1 \leq N \leq 10^5$

$1 \leq K \leq 10^9$

$1 \leq D \leq 5 \times 10^5$

$1 \leq P_i < A_i \leq 10^9$ for $1 \leq i \leq N$

Output

Output the maximum of games that Ian can buy using the budget of K dollars in D days.

Examples

input

```
3 10 6
10 10 10
3 4 5
```

output

```
2
```

input

```
3 10000 1
2 3 4
1 1 1
```

output

```
1
```

This page is intentionally left blank.

J Just Another FFT Problem

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
6TH · 13 AUG 2022

Recently, Justin have learned the FFT (Fast Fourier Transform) algorithm. To show his understanding of the related topics, he creates the following problem for you to solve.

You are given two strings S and T containing only lowercase English letters. The length of the two strings can be defined as $|S|$ and $|T|$, respectively. The indices of S and T start at 1. We use S_i to denote the i^{th} character of S and similarly T_i for the i^{th} character of T.

Justin defines a new operator $\otimes$, which can by applied to these two strings. The result of $S \otimes T$ is an array A of length $|S| + |T| - 1$. The value of A_i for $i = 1$ to $|S| + |T| - 1$ is:

$$A_i = \sum_{k=\max(1,i-|T|+1)}^{\min(i,|S|)} [S_k = T_{i-k+1}]$$

Here $[\dots]$ is the Iverson bracket. $[P]$ is defined to be 1 if P is true, and 0 if it is false.

Justin wants you to calculate the resulting array. For simplicity, he also gives you an integer M to compress the array, so you only need to output a single integer ans, which can be calculated by the following formula:

$$ans = \left(\sum_{k=1}^{|S|+|T|-1} A_k \cdot M^{k-1} \right) \mod 998244353$$

Input

The first two lines contain two strings, S and T ($1 \le |S|, |T| \le 5 \times 10^5$).

The third line contains one integer M ($1 \le M \le 10^6$).

Output

Output a single integer, ans.

Examples

input

```
puila
tiu
3
```

output

```
54
```

input

```
fft
justforfun
10
```

output

```
110210000
```

Note

In the first example, there are only two characters appear in both strings:

- The 2^{nd} character of S is equal to the 3^{rd} character of T
- The 3^{rd} character of S is equal to the 2^{nd} character of T

So $A_4 = 2$ and $ans = 2 \times 3^3 = 54$.

K Karaoke

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
6TH · 13 AUG 2022

In Karaoke, you are usually given a fixed amount of time to sing songs you like (e.g. 3 hours) and you can choose any songs that are available via the karaoke screen. Once time runs out, the screen will be automatically locked such that you can still finish your current song, but you would not be able to select any more songs nor proceed to the next songs.

In order to "extend" the singing time, people will usually choose the longest song available right before the screen is locked out so that they can still sing for the longest time after the screen is locked. In a popular karaoke chain, this song happened to be 勁歌金曲2 (For the sake of this task, let's call it "Golden Songs") by Leo Ku. The KTV of this song is 12 minutes and 48 seconds (or 768 seconds) long while a typical song is only about 3-5 minutes long. Therefore, if you pick Golden Songs as your last song right the screen is locked, instead of other songs, you would typically get to sing for at least an additional 7 minutes. Therefore, as a smart customer singing often in Karaoke, you look to maximize your singing time.

Formally, you have T seconds of singing time remaining ($1 \leq T \leq 18,000$). The screen will be automatically locked at the T-th second, meaning that the latest time to start a new song would be the $T-1$ second. In addition to Golden Songs, you will have an additional 3 songs that you can choose before the screen is locked. The three songs will be A, B, and C seconds long respectively ($180 \leq A, B, C \leq 300$). ALL songs can be played an unlimited number of times (as long as the screen is not locked) but must be played in full (i.e. it is not possible to play only part of the songs). For the sake of this task, assume that it will take no time to switch to the next song and there are currently no songs playing. Because you are too excited to sing, you would not leave any idle time in the karaoke (i.e. you must be playing a song at all times). Your task is to find out the maximum singing time possible.

Input

The only line contains four integers T, A, B, and C.

$1 \leq T \leq 18,000$

$180 \leq A, B, C \leq 300$

Output

Output a single integer – the maximum singing time possible in seconds.

Examples

input	output
601 180 210 300	1368

input	output
3 200 190 180	768

Note

In the first example, you should play the 3 additional songs (e.g. the first song once and the second song twice / the third songs twice) for a total of 600 seconds of singing time. 1 second before the screen locks, play Golden Songs for 768 seconds to achieve a total singing time of 1368 seconds.

In the second example, because there is not enough time left, the best strategy would be to play Golden songs right from the beginning for a total singing time of 768 seconds.

L Linear Game

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
6TH · 13 AUG 2022

Psychology has explained the fact that when people play the Rock-Paper-Scissors game, they tend to select moves repetitively. To experiment with this fact, Lucas has designed a Rock-Paper-Scissors game where each player always play the same move.

The game involves two teams, team A (with N players) and team B (with M players). Initially, all players are aligned on a straight line. All team A players are positioned on the left while all team B players are positioned on the right. Each player has a fixed move in the Rock-Paper-Scissors game, which is either rock, paper or scissors. All players do not change their moves throughout the game.

When the game starts, players will start moving continuously towards the other team, all with a common constant speed. Once two players meet head-to-head, they will play a Rock-Paper-Scissors game with their determined moves. The winner of the Rock-Paper-Scissors game can continue moving while the loser will be eliminated from the game. If there is a draw, the winner will be decided randomly.

You may have noticed that when all players from either team A or team B are eliminated, the remaining players will continue moving forever. Lucas is interested in investigating this scenario. Hence, he asks you to find the number of players from both teams who can **possibly** survive in the game and keep moving until the death of the universe.

In the Rock-Paper-Scissors game, Rock beats Scissors, Scissors beats Paper and Paper beats Rock. There is a draw if and only if two players make the same move.

Input

The first line consists of 2 integers, N and M, the number of players initially facing right and left respectively ($1 \leq N, M \leq 2 \times 10^5$).

The second line consists of a string of length $N + M$, the i-th character denotes the move of the i-th player. The players are numbered from left to right. It is guaranteed that the string consists of characters R, P and S only, representing rock, paper and scissors respectively.

Output

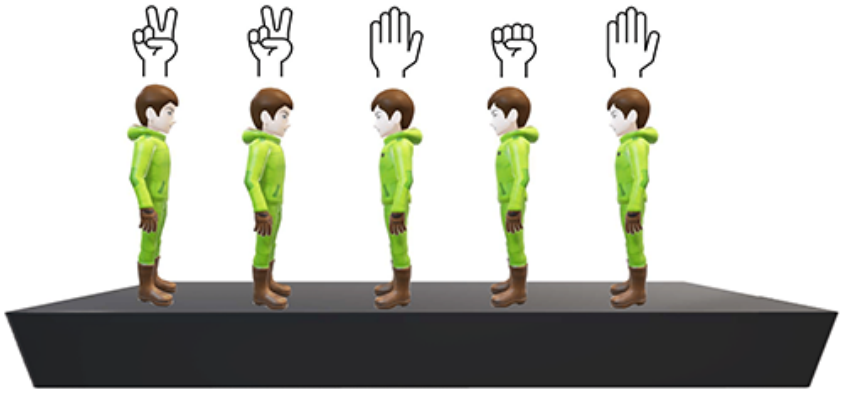
Output a single integer: the number of players that could possibly survive until the death of the universe.

Examples

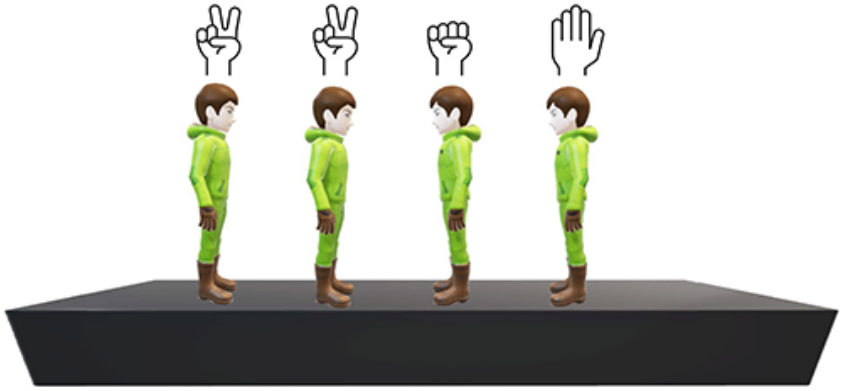
input	output
2 3 SSPRP	2
3 3 PRPSPR	3

Note

In the first sample, 2 players are initially facing the right and 3 players are initially facing the left. The following figure shows the initial positions of the players.

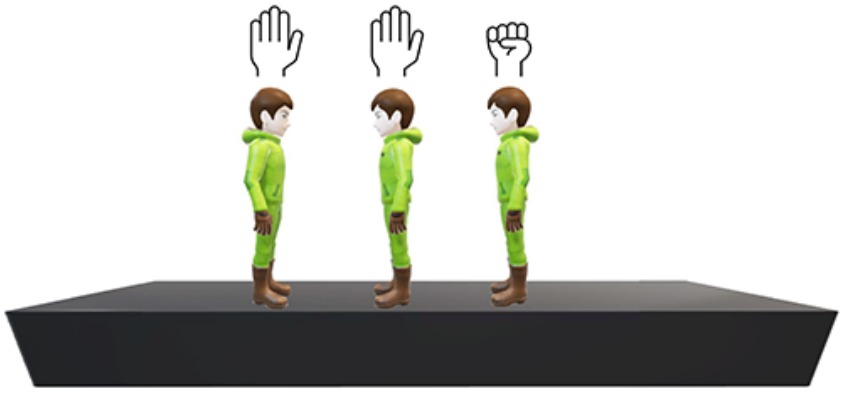


Then, the players start moving towards the other team. Player 2 (holding scissors) first meets player 3 (holding paper). As scissors beats paper, player 3 is eliminated from the game and player 2 can continue moving.



After 2 more rounds, we can conclude that player 4 (holding rock) and player 5 (holding paper) always survive. Thus, the answer is 2.

In the second sample, consider the following state:



If player 1 from team A wins, player 1 will survive eventually. Otherwise, if player 5 from team B wins, player 5 and player 6 will survive eventually. Therefore, the answer is 3.