TWGSS OI Team

2022/23 INTER-HOUSE CODING COMPETITION

INFORMATION

Date	6 July 2023	
Time	09:00 - 13:00	4 hours
Full Score	1000	

PROBLEMS

	Problem	Limit		Score	
		Time	*Memory*	*Subtasks*	*Total*
TH231	Mosaic Colour	1s	256MB	9 | 8 | 14 | 19	50
TH232	Elective Voting	1s	256MB	8 | 6 | 21 | 7 | 11 | 17	70
TH233	Good Knight	1s	256MB	13 | 7 | 24 | 36	80
TH234	Personalised Seating	1s	256MB	21 | 11 | 23 | 45	100
TH235	Attendance Game	1s	256MB	15 | 37 | 30 | 22 | 26	130
TH236	Dangerous Solution	1s	256MB	150	150
TH237	Left Alright	1s	256MB	34 | 17 | 40 | 19 | 44 | 26	180
TH238	Destined Gate	1s	256MB	18 | 26 | 36 | 29 | 22 | 39 | 29 | 41	240

REMINDERS

- All problems are divided into subtasks. All test cases in a subtask must be passed to get points.
- You may attempt the problems using C++20 or Python 3 (secondary language). It is not guaranteed that the problems are solvable using secondary language.
- Unless otherwise specified, your output will be automatically fixed as follows:
  - Trailing spaces in each line will be removed.
  - An end-of-line character will be added to the end of the output if not present.
- 64-bit integers may be required in some problems, use `long long` in C++ if needed.
- All stories are hypothetically written. In case of coincidences, they are coincidences.

TH231 Mosaic Colour

Score	Limit	
50	1s	256MB

"Please stand back from the train doors. DoDoDoDoDoDoDoDo…"

Just being fast enough to catch the train, you luckily found a seat and enjoyed the trip to your first-ever TWGOI *(The Wonderful Goodest Operation of Imaginary)* Training. Suddenly, you realised that you forgot to bring your phone outside! In *a day without mobile phones*, it would surely be a boring trip on the train.

You started to observe the decorations in the stations that your train had passed. You noticed that in every station, the wall is decorated with coloured Mosaic pattern. *Red… Green… Yellow… Green… Red… Blue…* You discovered that for every adjacent station, the colour of its Mosaic pattern is different. Having the memory of some stations, you decided to guess the colour of the remaining stations during this lame and boring trip on the train.

There are N stations in a service line. The i^{th} station should have the Mosaic colour C_i, represented by a capital letter (`A` to `Z`). For $1 \leq i \leq N-1$, $C_i \neq C_{i+1}$, and for $2 \leq i \leq N$, $C_{i-1} \neq C_i$. In other words, the colour is different for every adjacent station.

Unfortunately, as you are using coloured pencils to mark down the colour you guessed in your notebook, you only have a limited number of colours to choose from. In particular, you can only use no more than K colours to finish your answer, including those you remember and already filled in before guessing.

In this problem, the colours follow the alphabetical order, which starts with `A` and ends with `Z`. In other words, if $K = 5$, you can only use letters `A` to `E` to represent the colours.

INPUT

The first line consists of two integers N and K, separated by a space, representing the number of stations and the maximum number of colours that you can use.

The second line consists of a string S of length N, representing the colours of the stations. If the i^{th} character is `*`, the colour of the i^{th} station is uncertain and a guess is needed. Otherwise, it is C_i, i.e. the i^{th} station's colour and should not be changed.

Note that it is guaranteed that the given string S does not have equal consecutive characters that are not `*`.

OUTPUT

If there is no possible colour arrangements that satisfies all the requirements listed above, output `AC So Exciting` in the first and only line.

Otherwise, output a string of length N, with the i^{th} character being C_i, representing the colour of the i^{th} station.

If there are more than one possible valid arrangements, output any one of them.

SAMPLE TESTS

	Input	Output
1	7 21 *******	HUHRUOK

This sample satisfies the constraints of Subtask 1.

	Input	Output
2	8 8 E*DA*C**	EBDAFCGH

This sample satisfies the constraints of Subtask 2.

	Input	Output
3	9 2 BA*AB**AB	BABABABAB

This sample satisfies the constraints of Subtask 3.

	Input	Output
4	8 4 A*A**B*A	ABACABCA

Note that you do not have to use all of the first K alphabets.

	Input	Output
5	5 2 AB*A*	AC So Exciting

Although it is possible to put `B` into the 5^{th} station, it is both invalid to put `A` or `B` into the 3^{rd} station.

CONSTRAINTS

$1 \leq N \leq 2 \times 10^5$

$1 \leq K \leq 26$

S consists of capital letters that do not exceed the K^{th} alphabet or `*` only

SUBTASKS

	Points	Constraints
1	9	S consists of `*` only
2	8	$1 \leq N \leq K \leq 26$
3	14	$K = 2$
4	19	No additional constraints

TH232 Elective Voting

Score	Limit	
70	1s	256MB

"Welcome to the zeroth session of the training!"
"Zeroth?"
"Aren't you an OI person? We always start things with zero!"

Stepping into the room, you already knew that the training would have load of fun. *"Let us all start this revolutionary year with a vote!"* Ahh! The fun has already begun!

Trainees had precious opportunities to vote for their favourite topics. Of course, topics with higher votes would feature more in the training, while those with fewer votes would only be a minor part, or they would not appear in the training at all!

"Bong!" The clock strikes ten, the voting has ended, and the excitement of vote counting has begun. Among the V valid votes, the trainees had chosen their favourites among N topics. Excluding this zeroth session, there are Q sessions in the whole training this year. The number of sessions for each topic would be decided as follows, using Hare Quota:

- The Hare quota λ is calculated by dividing the total number of valid votes V by the total number of sessions Q. Mathematically, $\lambda = \frac{V}{Q}$. For the sake of simplicity, you may assume V **is always divisible by** Q, i.e. λ is an integer.
- For each topic, for each λ votes it got, a session is allocated to that topic. If a total of k sessions is allocated, $k\lambda$ votes would be subtracted from the valid vote of that topic.
- Finally, for the remaining session(s), they are assigned to topics following the descending order remaining valid votes, one by one topic. In other words, a topic with more remaining valid votes would have priority over a topic with a less remaining valid votes.
- In case of topics having the same remaining valid votes but there is not enough remaining sessions, in order not to upset their voters, extra sessions would be held for those topics.

"The voting results are as follows… Wait! The computer storing the results is automatically updating itself… Here's all data, can someone help me to code a program determining the results?" Wishing to leave an *impressive zeroth impression* on trainers, you are going to work on this!

INPUT

The first line consists of three integers N, Q and V, each separated by a space, representing the number of topics, the number of sessions and the number of total valid votes respectively.

The second line consists of N space-separated integers $A_1, A_2, \ldots, A_N$, with A_i representing the number of valid votes got by the i^{th} topic.

OUTPUT

Output N integers, each separated by a space, the i^{th} integer should be the number of sessions given to the i^{th} topic.

SAMPLE TESTS

	Input	Output
1	4 8 100000 41000 29000 17000 13000	3 2 2 1

In this situation, the Hare Quota $\lambda = \frac{V}{Q} = \frac{100000}{8} = 12500$.

- Topic `1`: 3 sessions is allocated, and $41000 - 3 \times 12500 = 3500$ valid votes remain.
- Topic `2`: 2 sessions is allocated, and $29000 - 2 \times 12500 = 4000$ valid votes remain.
- Topic `3`: 1 session is allocated, and $17000 - 12500 = 4500$ valid votes remain.
- Topic `4`: 1 session is allocated, and $13000 - 12500 = 500$ valid votes remain.

There is 1 remaining session. As topic `3` has the greatest number of remaining votes, topic `3` obtains the remaining session.

2	3 8 1000 125 375 500	1 3 4

This sample satisfies the constraints of Subtask 2.

3	5 5 75 10 10 10 20 25	1 1 1 1 2

In this situation, the Hare Quota $\lambda = \frac{V}{Q} = \frac{75}{5} = 15$.

- Topic `1`, `2`, `3`: no sessions is allocated, and 10 valid votes remain.
- Topic `4`: 1 session is allocated, and $20 - 15 = 5$ valid votes remain.
- Topic `5`: 1 session is allocated, and $25 - 15 = 10$ valid votes remain.

Even though there are only 3 remaining sessions, as topics `1`, `2`, `3` and `5` has the same greatest number of remaining votes, extra sessions will be held so that those topics would each end up with one more session.

CONSTRAINTS

$1 \le N \le 2 \times 10^5$

$1 \le Q \le 10^9$

$0 \le A_i \le 10^9$

$1 \le V = \sum_{i=1}^{N} A_i$

V is divisible by Q

SUBTASKS

	Points	Constraints
1	8	$1 \leq V \leq 10^9$ $Q = 1$
2	6	$1 \leq V \leq 10^9$ A_i is divisible by λ, i.e. the vote received by each topic is divisible by the Hare quota
3	21	$1 \leq V \leq 2.2 \times 10^6$ $1 \leq N \leq 82$ $1 \leq Q \leq 35$
4	7	$1 \leq N \leq 5000$
5	11	The situation where multiple topics having the same remaining valid votes but there is not enough remaining sessions would not happen
6	17	No additional constraints

TH233 Good Knight

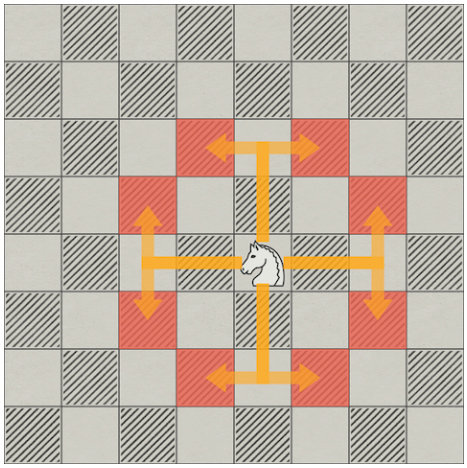
Score	Limit	
80	1s	256MB

"Good night!" The trainer ended the video call with his colleague in the opposite side of the globe with a good wish.

"Good ***Knight****? Thank you! Wait… How you know that I finally overthrow the* ***King*** *and beat the game?"* Turns out that Daniel is playing an online game called *"The Horsey Revenge"* during the TWGOI Training. Why? Because he is so smart!

In *"The Horsey Revenge"*, the player plays as the **Knight**, who got his girlfriend stolen by the **King**! The final target of the game is to take revenge on the **King** by getting the crown!

The game takes place in the *Chessington*, which is represented as an 8×8 chessboard. Initially, you, the **Knight**, are at cell (a_x, a_y). For each unit time, you can only move from opposite corners of an 1×2 rectangle to another. You cannot move out of the board, but you can move to position(s) that you previously moved to. There are cameras everywhere monitoring every movement by you, so you should not stay at a cell without taking a move as it is ultra dangerous!



The **King** will appear at cell (b_x, b_y) after T units of time before disappearing at the next moment in order to have a speed date with *your* girlfriend! Therefore the only way to defeat the **King** is to arrive at cell (b_x, b_y) at exactly time T.

"This game seems fun!" Unfortunately, as you did not bring your mobile phone to the training, and the computer there was not connected to the Internet, you could not play it! Therefore, you asked Daniel to generate some game cases for you in order to play the game.

Daniel suddenly realised, *"Oh no! Some of the cases are impossible to win! Maybe you have to write a checker yourself to determine whether it is possible to win the game with the cases I gave you."* Knowing that Daniel would be very angry if you do not play the game because of this when he already generated the game cases, you have no choice but to code a checker yourself.

INPUT

The first line consists of an integer T, the time that the **King** will appear.

The second line consists of two integers a_x and a_y, separated by a space, indicating the initial position of the **Knight**.

The third line consists of two integers b_x and b_y, separated by a space, indicating the appearing position of the **King**.

OUTPUT

Output `Good Knight` on the first and only line if the game is winnable, i.e. the **Knight** can defeat the **King**. Otherwise, output `God Save The King`.

SAMPLE TESTS

	Input	Output
1	1 1 3 3 2	Good Knight

The **Knight** can move to the **King**'s position in exactly 1 move.
$(1, 3) \to (3, 2)$

	Input	Output
2	3 1 3 3 2	Good Knight

The **Knight** can move to the **King**'s position in exactly 3 moves.
$(1, 3) \to (3, 4) \to (5, 3) \to (3, 2)$

	Input	Output
3	2 1 1 3 2	God Save The King

The **Knight** can move from cell $(1, 1)$ to cell $(3, 2)$ in 1 move. However, as the **Knight** must perform 2 moves, it is impossible for the **Knight** to revenge.

	Input	Output
4	1000000000 1 1 7 5	Good Knight

CONSTRAINTS

$0 \leq T \leq 10^9$

$1 \leq a_x, a_y, b_x, b_y \leq 8$

SUBTASKS

	Points	Constraints
1	13	$0 \leq T \leq 1$
2	7	$(a_x, a_y) = (b_x, b_y)$
3	24	$0 \leq T \leq 10$
4	36	No additional constraints

TH234 Personalised Seating

Score	Limit
100	1s | 256MB

"As this is our zeroth session, let us play an exciting ice-breaking game!"

"Game…game? It… Oh no! It is horrible!" Fearing the embarrassment talking with strangers, Jack's *hands were shaking 6.6 times a second* and his brain stopped working.

"Can anyone help us to arrange the seating plan?" a trainer asked.

Being Jack's *"best"* friend, you decided to make a seating plan which people with completely different personality sitting next to each other.

The seats in the room are arranged in tables of two. There is a total of N trainees, each with different **outgoingness**. Let the **outgoingness** of two people sitting in the same table be x and y respectively, their **embarrassment value** is defined to be $|x-y|$, i.e. the difference in their **outgoingness**. If only one person is sitting in a pair of seat, he/she will not get embarrassed, and the **embarrassment value** is 0. The target for your *specialised* seating plan is to **maximise** the **total embarrassment value**, summing over all the tables.

"I can help!" you shouted immediately.

"Ok, please write a code to do so then!" Oh no! You fell into the trap. Your *hands were shaking 6.6 times a second* and your brain stopped working. You have to do it anyways.

"Hi! Can you please make a seating plan that I can sit with my friends?" Jack came near and asked.

"Ehh… Of course… I'll send you the plan once I get it finished." you replied. You decided to make a fake seating plan that **minimise** the **total embarrassment value** for sending it to Jack in order to extend your friendship a bit longer before you hand in the real, evil version to the trainers.

INPUT

The first line consists of an integer N and a character c, separated by a space.

If $c =$ `R`, the real seating plan should be made; if $c =$ `F`, the fake seating plan should be made.

The second line contains N integers $A_1, A_2, \ldots, A_N$, each separated by a space, with A_i representing the **outgoingness** of the i^{th} person.

OUTPUT

If $c =$ `R`, output an integer, the **maximum total embarrassment value** among all pairs of seats.

If $c =$ `F`, output an integer, the **minimum total embarrassment value** among all pairs of seats.

SAMPLE TESTS

	Input	Output
1	5 R 53 54 67 7 9	105

- Two people with **outgoingness** 67 and 9 sit next to each other gives a **embarrassment value** of $67 - 9 = 58$
- Two people with **outgoingness** 54 and 7 sit next to each other gives a **embarrassment value** of $54 - 7 = 47$
- No one would sit next to the person with **outgoingness** 53.

Therefore the **total embarrassment value** is $58 + 47 = 105$, which can be proved as **maximum**.

2	6 F 18 -31 45 -26 -7 26	49
3	5 F 53 54 67 7 9	3

- Two people with **outgoingness** 53 and 54 sit next to each other gives a **embarrassment value** of $54 - 53 = 1$
- Two people with **outgoingness** 9 and 7 sit next to each other gives a **embarrassment value** of $9 - 7 = 2$
- No one would sit next to the person with **outgoingness** 67.

Therefore the **total embarrassment value** is $1 + 2 = 3$, which can be proved as **minimum**.

CONSTRAINTS

$1 \leq N \leq 2 \times 10^5$

$c =$ `R` or `F`

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

SUBTASKS

	Points	Constraints
1	21	$c =$ `R`
2	11	$c =$ `F` N is even
3	23	$c =$ `F` $1 \leq N \leq 5000$
4	45	$c =$ `F`

TH235 Attendance Game

Score	Limit	
130	1s	256MB

"The attendance game is going to start! Are you guys ready?"
"Why you don't do announcement using Chinese? That's unfair!"
"Can someone go help him?"

Being zeroth session of the training, a brand-new attendance game is introduced to trainees. By taking enough attendance in the game throughout the training, trainees would get a shiny certificate!

The attendance game runs as follows:
There are N participants numbered 1 to N. The host would show M cards on the screen, each with a capital letter (`A` to `Z`) on it, while some letters may repeat or not appearing at all. Then each participant has to match letters following the order shown on the screen using his/her device. There are four operations for each of the participants to do:

Format	Operation
`c x`	The letter c would be clicked x times, which means x letter c would be added to the end of the participant's input.
`delete x`	The last x letters would be deleted from the participant's current input. It is guaranteed that the original input is at least x letters long.
`clear`	The participant's input would be be cleared.
`submit`	The participant's input is submitted for checking. If the input is exactly the same as the letters shown on the screen with the correct order, the participant would finish the game and his/her attendance would be recorded. Otherwise, the input is considered incorrect, and it would be cleared. Note that a submitted input with length exceeding M will also be considered incorrect.

Once the game is finished by a participant, any further input by him/her would simply be ignored.

As usual, trainees in the training are very competitive! They always want to be the fastest to finish the game and get their attendances recorded. In fact, they will not get anything extra by finishing this fast *(except fame, maybe)*.

"Seems you guys are not yet ready… Maybe this… This may help you understand the rules better. Don't worry! This only depends on your problem solving skills. Therefore it is completely fair! You are given all Q operations from all N participants in a trial run among trainees in chronological order (ascending order of time). And you are given the task to generate a list of successful participants according to their time finishing the game.

INPUT

The first line consists of three integers N, M and Q, each separated by a space, representing the number of participants, the number of cards shown on the screen and the total number of operations by all participants respectively.

The second line consists of a string S of length M, representing the cards shown on the screen with the correct order.

The i^{th} line of the next Q lines consists of an integer p_i and an operation in one of the format shown above, representing the i^{th} operation in chronological order done by the participants numbered p_i.

OUTPUT

If no participants finished their games, output `Empty Class` in the first and only line.

Otherwise, output two lines.

The first line should contain an integer K, the number of participants who finished their games.

The second line should contain K integers, each separated by a space, with the i^{th} integer being the ID of the participant who finished i^{th} (the i^{th} fastest) in the game.

SAMPLE TESTS

Input	Output
3 8 8 AAAAAAAA 1 A 5 3 A 7 1 submit 3 A 2 3 delete 1 3 submit 1 A 8 1 submit	2 3 1

This sample satisfies the constraints of Subtask 1.

For the 3^{rd} operation, participant `1`'s input is `AAAAA`, which is incorrect, and is cleared.
For the 6^{th} operation, participant `3`'s input is `AAAAAAAA`, which is the same as the cards shown on the screen with the correct order. Therefore participant `3` is the first one to finish the game.

2

2 1 7 W 2 H 1 1 E 1 1 clear 2 W 1 2 submit 1 W 1 1 submit	1 1

This sample satisfies the constraints of Subtask 4.

3

1 2 4 AB 1 A 1 1 B 1 1 C 1 1 submit	Empty Class

This sample satisfies the constraints of Subtask 3.

CONSTRAINTS

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

$1 \leq Q \leq 3 \times 10^5$

S consists of capital letters (`A` to `Z`) only

$1 \leq p_i \leq N$

c is a capital letter (`A` to `Z`)

$1 \leq x \leq M$

SUBTASKS

	Points	Constraints
1	15	S consists of `A` only $c =$ `A`
2	37	$1 \leq N, M, Q \leq 2000$
3	30	There is no `delete` or `clear` operations
4	22	$x = 1$ for all valid operations, i.e. only one letter will be added or deleted for each respective operation
5	26	No additional constraints

TH236 Dangerous Solution

Score	Limit	
150	1s	256MB

BOOOOOOOOMMMM!!!! Timothy's eyes nearly popped out of his head.

It was nearly the end of the zeroth session of the TWGOI training. Timothy was sleeping and the trainers seems to be already so tired of preparing the slides last night and no one told him to wake up. Timothy has entered the dream of competing in the Super Science Experiments for Eager Youngsters.

Professor Snookhorn informed Timothy about the items on the bench. There are N test tubes of solution, each consists of either of two types: type `A` (for "acidic") and type `B` (for "basic"). However, he did not know which type of the solution contained by each test tube. It is known that no reaction would occur if solutions of the same type is put together. In other words, there would be no reaction between two type `A` solutions, same for two type `B` solutions. However, if he adds type `A` and type `B` solutions together, an explosion would occur. Specifically, if he add c_A units of type `A` solution and c_B units of type `B` solution, an explosion of strength $\min(c_A, c_B)$ would occur.

"So that's all the information for this year's Team Formation Test, any questions?" You suddenly woke up. *"Oh no!"* Knowing that Timothy had never listened in Chemistry lessons, you realised that an explosion would certainly occur if Timothy starts to handle the solutions. *"Wake up! Wake up!"* You woke Timothy up and forced him to leave the dream. You decided to quickly fall asleep again, hoping that you can sneak into the dream again and prevent the famous explosion from happening!

"The Super Science Experiments for Eager Youngsters begins!" You successfully entered the dream! In front of you are two waste bottles. You are going to dispose all solution in all test tubes into the two bottles without causing *major* explosion, which will occur if you mix a whole test tube of type `A` solution and a whole test tube of type `B` solution together.

Luckily, you can conduct several trial experiments to determine how you can dispose the solution. In each trial, you can add a certain volume of solution from several (possibly one) test tubes to a container. An anti-explosive agent is added so that no explosion immediately occurs. After a button is pressed, the effect of the anti-explosive agent is cancelled by some magical chemical reaction. The strength of the trial explosion is then recorded by an explo-metre. The trial mixture is cleared upon pressing the button.

Of course this is easily doable with a lot of trials. Unfortunately, Timothy would go asleep again very soon and cause the explosion in the dream to occur! You have no options but to dispose the solution quick enough, using the minimum number of trials possible.

IMPLEMENTATION

You should implement the function `void handleWaste(int N)`.

You can call the following functions:

`void addLiquid(int tube_id, int volume)`

- $1 \leq \text{tube\_id} \leq N$
- $1 \leq \text{volume} \leq 1024$
- At most once per tube_id per trial.

`int pressButton()`

- This function returns the strength of the explosion occured.
- The total number of calls must not exceed 1000.

`void pourWaste(int tube_id, int waste_id)`

- $1 \leq \text{tube\_id} \leq N$
- $1 \leq \text{waste\_id} \leq 2$
- This function should be called exactly once for each test tube, i.e. for $1 \leq \text{tube\_id} \leq N$.

TEMPLATE

```
#include <bits/stdc++.h>
#include "solution.h"

using namespace std;

void handleWaste(int N){
    // Write your code here...
    addLiquid(1, 1);
    addLiquid(2, 1);
    int b = pressButton();

    pourWaste(1, 1);
    pourWaste(2, 1);
    return;
}
```

SCORING

On each test case, you will receive zero score if your program:

- does not exit correctly, or
- violates any rules mentioned in the section **"Implementation"**, or
- pour two different types of solution to the same waste bottle.

In this problem, you can obtain a partial score. Your score will depend on K, the number of times `pressButton()` is called.

$$\text{score} = \begin{cases} 100\% & \text{if } K \leq 88 \\ [85 + 5(91 - K)]\% & \text{if } 88 < K \leq 91 \\ [67 + 2(100 - K)]\% & \text{if } 91 < K \leq 100 \\ [20 + 0.0522(1000 - K)]\% & \text{if } 100 < K \leq 1000 \\ 0 & \text{otherwise} \end{cases}$$

Your score on a subtask is the lowest score you get among all test cases.

SAMPLE GRADER

The sample grader reads the input in the following format:

- Line 1: N
- Line 2: $C_1, C_2, \ldots, C_N$, which C_i is either `A` or `B`, representing the type of solution contained in the i^{th} tube

If the sequence of instructions is incorrect, or it gives an incorrect output for any of the test cases, the sample grader outputs a line `Wrong Answer: [Reason]`. Otherwise, it outputs a line `Accepted: K trials used`.

If an invalid input is supplied to the sample grader, the behaviour of the grader is undefined.

Compilation Command: `g++ sample_grader.cpp solution.cpp -o solution`

Run Command: `./solution` (Linux or Mac) OR `solution` (Windows)

SAMPLE TESTS

Input	Output

3
ABA
	addLiquid(1, 2);
	addLiquid(2, 3);
	pressButton(); // returns min(2, 3) = 2
	
	addLiquid(1, 2);
	addLiquid(3, 1);
	pressButton(); // returns min(3, 0) = 0
	
	pourWaste(1, 1);
	pourWaste(2, 2);
	pourWaste(3, 1);

CONSTRAINTS

$2 \leq N \leq 1000$

TH237 Left Alright

Score	Limit
180	1s | 256MB

"Go left?"
"Right."
"Ok, let's go left then."
"All right!"
"Alright! Let's go left!"

Go left? Or go right? This has always been one of the hardest question left for all the *right honourable friend* like you to solve. The zeroth session of the TWGOI training has ended, you and your friends decided to walk and find things to eat. For every intersection point, your group would play a game called *"Left Alright"* to determine whether the next move is left or right.

There are N friends walking with you. The *"Left Alright"* game is *outright straightforward*. Each friend would have a preference of answering a direction, either `L` (left) or `R` (right) upon receiving the initial direction from you or the direction from previous person's answer. In other words, friend i would always answer L_i when receiving `L`, and answer R_i when receiving `R` from you or friend $i-1$.

There are two situations that your group would stop and perform the following operations:

Type	Format	Operation
Update	$f\ l\ r$	Friend f change his/her mind and decide to say l upon receiving `L`, and say r upon receiving `R`. Formally, L_i would change to l, R_i would change to r.
Game	$d\ x\ y$	The group arrive at an intersection point, and would play the *"Left Alright"* game. The game would start with the friend x receiving the initial direction d decided by you and the result would be what the answer of friend y says in the end.

Knowing your friends' preference so well, you are going to write a program to simulate this *young, simple and naïve* game and get the result without having to wait so long when encountering every intersection.

INPUT

The first line consists of two integers N and Q, separated by a space, representing the number of friends and the number of operations respectively.

The i^{th} line of the next N lines each consists of two characters, L_i and R_i, separated by a space. Friend i will answer L_i upon receiving `L`, and answer R_i upon receiving `R`.

The j^{th} line of the next Q lines consists of an integer t_j and an operation. If $t_j = 1$, an **"Update"** operation follows. If $t_j = 2$, a **"Game"** operation follows.

OUTPUT

For each **"Game"** operation, output the final direction of the game, either `L` or `R`.

SAMPLE TESTS

	Input	Output
1	3 5 L L L R R L 2 L 1 3 1 2 L R 1 3 L R 2 L 1 2 2 R 2 3	R L R

CONSTRAINTS

$1 \le N, Q \le 2 \times 10^5$

$L_i, R_i, l, r =$ `L` or `R`

$t_j =$ `1` or `2`

$1 \le f \le N$

$1 \le x \le y \le N$

SUBTASKS

	Points	Constraints
1	34	$1 \le N, Q \le 5000$
2	17	$x = 1$ for every **"Game"** operation Any **"Update"** operations are before any **"Game"** operations
3	40	$L_i \ne R_i$ for $1 \le i \le N$ $l \ne r$ for every **"Update"** operation There are at most 10 **"Update"** operations
4	19	$L_i \ne R_i$ for $1 \le i \le N$ $l \ne r$ for every **"Update"** operation
5	44	There are no **"Update"** operations
6	26	No additional constraints

TH238 Destined Gate

Score	Limit
240	1s | 256MB

Observing the Mosaic pattern of train stations…
Voting for your favourite topic…
Taking revenge on the King…
Sitting with a stranger…
Playing the ultra fun attendance game…
Handling the dangerous solutions…
Going left, right, left and right…

Many events happened in a busy day, and now you are going to… decide your own ending!

A **world line segment** is defined to be a cause-and-effect relationship between two **spacetime points**. If a **world line segment** existed from **spacetime point** A to **spacetime point** B, that means it is **possible** to go from A to B. There can be multiple **world line segments** starting from each **spacetime points**.

In the world, here are in total N **spacetime points**. A **spacetime point** is called an **event** if there is one or more **world line segments** outcoming from it, and an **ending** otherwise. The first $N-K$ **spacetime points** are events while the last K are **endings**. The 1^{st} spacetime point is the **initial event**, which all events and endings are reachable from the initial event by following one or more **world line segments**.

At each **event**, you may choose to follow one of the **world line segments** starting from it by changing your action accordingly. The next **spacetime point** always comes later than the current **spacetime point**, so it is impossible to visit the same **spacetime point** twice by following **world line segments** normally.

You feel this is unfair! *Why someone else is so success, and I was just here to suffer?* You wonder. As a hacker, you eventually meet a Mad Scientist who figure out the way to revert in time - inventing the **Time Leap Machine**, which can be used to go from an **event** to other **past event**, for at most T hours. The **Time Leap Machine** could be available to you to use at some **event(s)**. However, at some **events**, it might be impossible to use the **Time Leap Machine** for various reasons – it might not yet been invented, it is broken at that moment, or you are simply too far away from the Laboratory with the Machine. **You can only Time Leap to events that you have been. For the sake of simplicity, for each event, there exists a unique series of world line segments from the initial event to this event.**

You, at a specific event, wish to reach the **Destined Gate** – one of the many possible endings that you truly desire. As a fellow hacker, you decide to use your hacking ability to find, for each situation, the minimum number of **Time Leaps** required, or tell it is impossible to reach the **Destined Gate**.

INPUT

The first line consists of five space-separated integers N, K, M, Q and T, representing the number of events, the number of endings, the number of world line segments, the number of situations and the number of hours you can go back to using the Time Leap Machine respectively.

The second line consists of $N-K$ space-separated integers $c_1, c_2, \ldots, c_{N-K}$. If $c_i = 0$, you cannot use the Time Leap Machine at the i^{th} event, while if $c_i = 1$, you can use the Time Leap Machine at the i^{th} event.

The i^{th} line of the next M lines consists of three space-separated integers u_i, v_i and t_i, describing the i^{th} world line segment, which means after event u_i, it is possible that the following spacetime point is spacetime point v_i, after t_i hours.

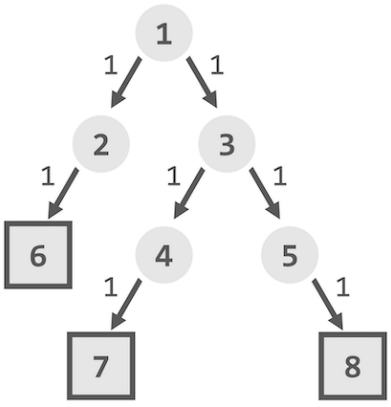
The j^{th} line of the next Q lines consists of two space-separated integers x_j and e_j, describing the j^{th} situation with x_j being the event you currently at and e_j being the desired Destined Gate.

OUTPUT

For each situation, output `-1` if it is impossible to reach the **Destined Gate**. Otherwise, output the minimum number of **Time Leaps** required on a single line.

SAMPLE TESTS

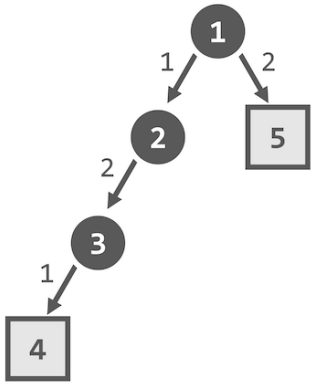
Input	Output
8 3 7 3 1 0 0 0 0 0 1 2 1 1 3 1 2 6 1 3 4 1 3 5 1 4 7 1 5 8 1 2 6 3 7 5 6	0 0 -1



This sample satisfies the constraints of Subtasks 1 and 6.

2

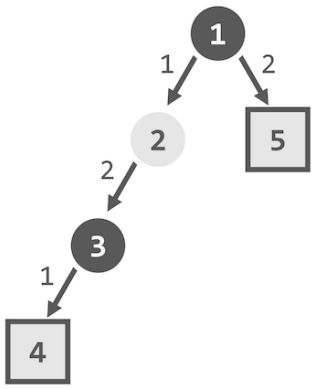
5 2 4 4 2 1 1 1 1 2 1 2 3 2 3 4 1 1 5 2 2 4 1 5 2 5 3 5	0 0 1 2



This sample satisfies the constraints of Subtasks 2, 4 and 5.

3

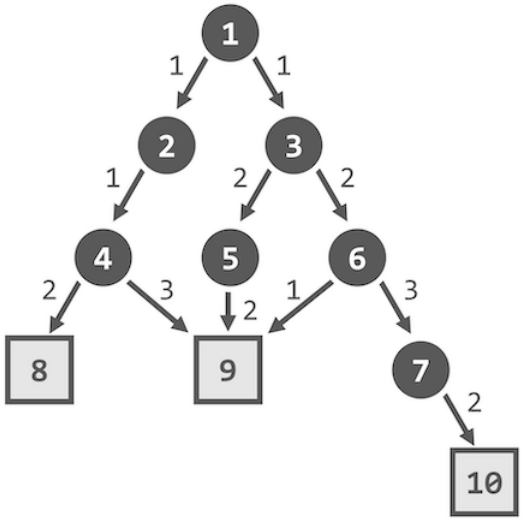
5 2 4 4 2 1 0 1 1 2 1 2 3 2 3 4 1 1 5 2 2 4 1 5 2 5 3 5	0 0 -1 -1



This sample satisfies the constraints of Subtask 5.

4

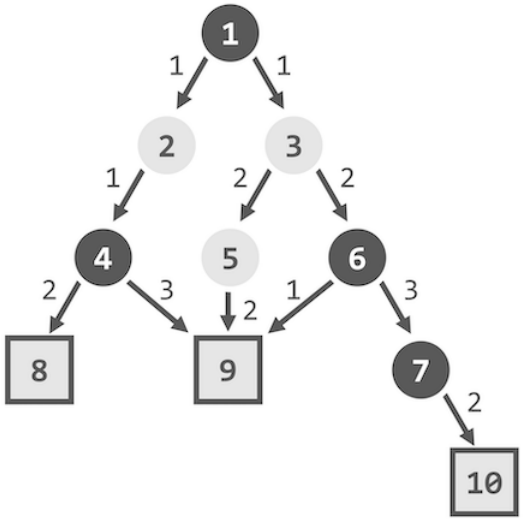
10 3 11 5 2 1 1 1 1 1 1 1 1 2 1 1 3 1 2 4 1 4 8 2 4 9 3 5 9 2 3 5 2 3 6 2 6 9 1 6 7 3 7 10 2 2 8 4 10 3 8 7 9 5 8	0 1 1 -1 2



This sample satisfies the constraints of Subtask 2.

5

10 3 11 5 2 1 0 0 1 0 1 1 1 2 1 1 3 1 2 4 1 4 8 2 4 9 3 5 9 2 3 5 2 3 6 2 6 9 1 6 7 3 7 10 2 2 8 4 10 3 8 7 9 5 8	0 1 -1 -1 -1



CONSTRAINTS

$2 \le N \le 10^5$

$1 \le K \le N - 1$

$1 \le M \le 2 \times 10^5$

$1 \le Q \le 10^5$

$1 \le T \le 10^9$

$0 \le c_i \le 1$

$1 \le u_i, v_i \le N$

$(u_i, v_i) \ne (u_j, v_j)$ for $i \ne j$

$1 \le t_i \le 5 \times 10^3$

$1 \le x_j \le N - K < e_j \le N$

SUBTASKS

	Points	Constraints
1	18	$1 \le Q \le 10$ $c_i = 0$ for $1 \le i \le N - K$
2	26	$1 \le N, M, Q \le 300$ $c_i = 1$ for $1 \le i \le N - K$
3	36	$1 \le N, M, Q \le 5000$
4	29	$K = 2$ $M = N - 1$ $u_i = i, v_i = i + 1$ for $1 \le i \le N - 2$ $u_{N-1} = 1, v_{N-1} = N$ $c_i = 1$ for $1 \le i \le N - K$
5	22	$K = 2$ $M = N - 1$ $u_i = i, v_i = i + 1$ for $1 \le i \le N - 2$ $u_{N-1} = 1, v_{N-1} = N$
6	39	$c_i = 0$ for $1 \le i \le N - K$
7	29	$1 \le K \le 10$
8	41	No additional constraints