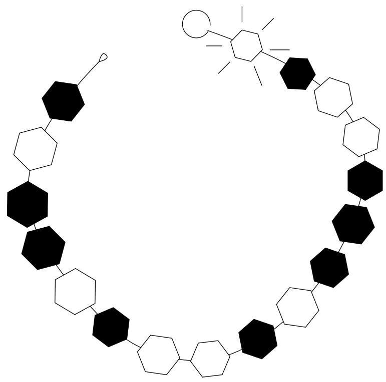
Input File: `pearls.in`
Output File: –
Source Code: `pearls.pas/.c/.cpp`

100 Points
Time Limit: 2 s
Memory Limit: 16 MB

The Pearl Necklace

There are two dwarf clans living in a mountain: the red dwarves and the green dwarves. During a joint expedition, a group of green and red dwarves finds a necklace chain consisting entirely of worthless black and white glass pearls except for a valuable diamond at the end of the chain.



Both dwarf clans want to take possession of the diamond. The dwarves decide to resolve the situation in a friendly manner and play the following game:

Each of the N dwarves is assigned a unique ID number from 1 through N. Each dwarf is given two publicly-known lists of dwarf ID numbers, a white list and a black list. (The dwarves' lists may be different from each other.) Each list may contain ID numbers of both red and green dwarves. During the game, the necklace is passed around according to the following rules: When a dwarf receives the necklace, he removes the first pearl. If this pearl is white, he hands the remainder of the necklace to a dwarf of his choice on his white list (which could be himself). If this pearl is black, the remainder goes to a dwarf of his choice on his black list (which also could be himself). To start the game, the dwarf who receives the necklace first is determined randomly.

Eventually, the necklace will consist of only the diamond. The dwarf who receives the necklace in this state keeps it for his clan, and the game is over.

Write a program that helps the green dwarves obtain the diamond. Use the library described below. You may assume that the red dwarves play optimally.

Library

You are provided with a library that contains the following functions:

- `getNext()` must be called when it is a red dwarf's turn. It returns the ID number of the dwarf to whom the red dwarf hands the necklace.
- `setNext(d)` must be called when it is a green dwarf's turn. The parameter d specifies the ID number of the dwarf to whom the green dwarf will hand the necklace.

- `finish()` must be called when the game is over. It will terminate your program.

For debugging your program, test versions of the library are provided. You will find the library source files `pearls_lib.h` and `pearls_lib.pas` in the directory `/home/ceoi/` or `c:\ceoi`. In this version of the library, the red dwarves always pass the necklace to the first dwarf on the appropriate list.

Specification for C/C++ programmers

Use the directive `#include "pearls_lib.h"` to include a library that provides the following functions:

```
int getNext(void);
void setNext(int d);
void finish(void);
```

Specification for Pascal programmers

Use the statement `uses pearls_lib;` to include a library that provides the following functions:

```
function getNext:Integer;
procedure setNext(d:Integer);
procedure finish;
```

Input

The first line of the input file `pearls.in` contains the initial length L of the necklace ($1 \leq L \leq 1000$), the number N of dwarves ($1 \leq N \leq 1000$), as well as the ID number F of the dwarf who gets the necklace first ($1 \leq F \leq N$). Note that $1 \leq i \leq N$ for any dwarf ID number i.

The second line contains L characters describing the necklace. Each of the first $L - 1$ characters is either the letter B or the letter W. B represents a black pearl, and W represents a white pearl. The last character is the letter D, representing the diamond.

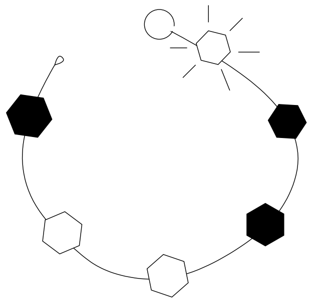
The subsequent N lines describe the dwarves. The ith of these lines describes the dwarf with ID number i. Each of these lines starts with a number specifying the color of the dwarf: 0 for a green dwarf, 1 for a red dwarf. Next comes the length L_B of the dwarf's black list ($1 \leq L_B \leq 20$), followed by the dwarf ID numbers on the black list. This list is followed by the length L_W of the dwarf's white list ($1 \leq L_W \leq 20$) and the L_W dwarf ID numbers on the white list.

Output

None

Example

pearls.in	Library Calls
`6 4 2`	`setNext(1)`
`BWWBBD`	`setNext(4)`
`0 1 2 1 4`	`getNext() -> 1`
`0 2 1 3 1 1`	`setNext(2)`
`1 1 4 1 4`	`setNext(1)`
`1 2 2 3 1 1`	`finish()`



Grading

If you do not call `finish()`, or if you call `setNext(d)` even though it is not a green dwarf's turn, or if dwarf d is not on the list, or if you call `getNext()` even though it is not a red dwarf's turn, you will receive a score of 0. You will also score 0 if at the time you call `finish()`, a red dwarf is in possession of the diamond or if the necklace contains more than just the diamond. Only if a green dwarf keeps the diamond at the end will you receive a perfect score. Each test case is designed in such a way that this can be achieved.