Problem	LIST	CARDS	MONKEY
Executable file	LIST.EXE	CARDS.EXE	MONKEY.EXE
Source file	LIST.BAS LIST.PAS LIST.C LIST.CPP	CARDS.BAS CARDS.PAS CARDS.C CARDS.CPP	MONKEY.BAS MONKEY.PAS MONKEY.C MONKEY.CPP
Input file	LIST.IN	CARDS.IN	MONKEY.IN
Output file	LIST.OUT	CARDS.OUT	MONKEY.OUT
Time limit per test	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10
Points per test	5	7	8
Total points	**50**	**70**	**80**
	200		

LIST

Little Michael lives in a small country and likes to watch a TV music show which is emitted every Sunday afternoon. It presents the same songs every week and, according to votes, displays popularity list of those songs.

Michael played too long with his friend one Sunday and failed to see new popularity list. He was sad, but he soon realized that he would be able to at least partially reconstruct the popularity list next week. Apart from the positions of songs, a list contains information about change of positions of all songs with regard to their last week's positions. More precisely, each song is given a mark indicating whether that song stayed at same position, gained or lost popularity since last week.

Write a program that will using given popularity list help Michael to determine **one** possible last week's popularity list.

Input data

The first line of input file contains an integer N, $1 \leq N \leq 100$, number of songs on popularity list.

Next N blocks describe a popularity list. A block consists of two lines. First line of i^{th} block contains name of i^{th} song. A name of a song consists of at most 100 uppercase letters of English alphabet. Second line of a block contains one of three words: **UP** (a song went up on a list), **DOWN** (a song went down on a list) or **SAME** (position stayed the same), describing a change on a list with respect to last week's list.

Output data

The output file contains one possible popularity list from last week in N lines.

Each line should contain name of a song so that i^{th} line contains name of i^{th} song on a list.

Note: A solution needs not to be unique, but there always will be at least one solution for each test data.

Examples

LIST.IN

```
4
LOVE
UP
WINE
UP
WATER
DOWN
HATE
DOWN
```

LIST.OUT

```
WATER
HATE
LOVE
WINE
```

LIST.IN

```
4
HOTDOG
UP
HAMBURGER
DOWN
ICECREAM
SAME
FRENCHFRIES
DOWN
```

LIST.OUT

```
HAMBURGER
FRENCHFRIES
ICECREAM
HOTDOG
```

LIST.IN

```
5
HIGHHOPES
UP
LOWFEELINGS
UP
UPANDDOWN
DOWN
IAMSTILLSTANDING
DOWN
FOOLINGAROUND
DOWN
```

LIST.OUT

```
UPANDDOWN
IAMSTILLSTANDING
FOOLINGAROUND
HGHHOPES
LOWFEELINGS
```

CARDS

Dave's little son Maverick likes to play card games, but being only four years old, he always lose when playing with his older friends. Also, arranging cards in his hand is quite a problem to him. When Maverick gets his cards, he has to arrange them in groups so that all the cards in a group are of the same color. Next, he has to sort the cards in each group by their value – card with lowest value should be the leftmost in its group. Of course, he has to hold all the cards in his hand all the time.

He has to arrange his cards as quickly as possible, i.e. making as few moves as possible. A move consists of changing a position of one of his cards.

Write a program that will calculate the lowest number of moves needed to arrange cards.

Input data

The first line of input file contains two integers C, number of colors ($1 \leq C \leq 4$), and N, number of cards of the same color ($1 \leq N \leq 100$), separated by a space character.

Each of the next C*N lines contains a pair of two integers X and Y, $1 \leq X \leq C$, $1 \leq Y \leq N$, separated by a space character. Numbers in each of those lines determine a color (X) and a value (Y) of a card dealt to little Maverick. The order of lines corresponds to the order the cards were dealt to little Maverick.

No two lines describe the same card.

Output data

The first and only line of output file should contain the lowest number of moves needed to arrange the cards as described above.

Examples

CARDS.IN

```
2 2
2 1
1 2
1 1
2 2
```

CARDS.OUT

```
2
```

CARDS.IN

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

CARDS.OUT

```
0
```

CARDS.IN

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

CARDS.OUT

```
2
```

MONKEY

A hungry monkey wants to eat banana. The monkey and banana are in a labyrinth consisting of rooms and corridors connecting them. Rooms can be in one of two states: locked or unlocked. If a room is locked, then the monkey cannot enter that room, but can leave it. Unlocked rooms can be freely entered and left.

Some rooms contain a switch. Pressing a switch changes states of a group of rooms, i.e. locked rooms from that groups become unlocked and vice versa. Same switch always changes states of the same group of rooms.

Upon entering a room having a switch, the monkey can press it if he chooses to do so.

Write a program that will help the monkey to find the room with banana as soon as possible, i.e. to find the smallest number of corridors the monkey has to pass to find the room with banana, possibly by pressing some of the switches.

Input data contain description of rooms and corridors in the labyrinth, initial states of all rooms, a list of switches and for each switch a list of rooms whose states can changed by it.

Input data

The first line of input file contains two integers N, total number of rooms ($1 \leq N \leq 100$), and S, number of switches, i.e. number of rooms having switches, in labyrinth ($1 \leq S \leq 8$). Switches are located in the rooms with numbers from 1 to S.

The next N lines contain descriptions of rooms. The room number i is described with $(i+1)^{th}$ line which begins with either **0** (if the corresponding room is initially unlocked) or **1** (if initially locked). An integer K follows, total number of rooms connected via corridors with the room being described by that line. After it follows K numbers denoting those K rooms. A space character separates successive numbers in same line.

The following S lines describe switches, from first to S^{th} switch.

Each of those lines begins with an integer L, the number of rooms in a group whose states can be changed by switch being described. Following are L numbers – numbers of rooms in the group.

The last line of input file contains two numbers A and B; A is number of a room where the monkey starts its search for banana and B is number of room containing banana.

Output data

The first and only line of output file should contain the smallest number of corridors the monkey has to pass to find banana.

Note: Each test data will have a solution, i.e. there will always be a way to come to room B from room A.

Examples

MONKEY.IN

```
4 1
0 1 3
1 2 3 4
0 2 1 2
0 1 2
1 2
3 4
```

MONKEY.OUT

```
4
```

MONKEY.IN

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

MONKEY.OUT

```
3
```

MONKEY.IN

```
6 2
0 2 6 5
1 2 4 6
0 1 4
1 3 2 5 3
0 3 1 4 6
0 3 1 5 2
3 2 5 3
1 4
6 3
```

MONKEY.OUT

```
8
```