Problem 6 - Package Arrangement

Time Limit	:	3 s
Memory Limit	:	1024 MB

Problem Description

Ling is a worker in a factory. Each day, a sequence of packages, each of a different height, are "pushed" to several production lines. Each production line can be viewed as a queue where the packages that are "pushed" into the line earlier are in its front.

Ling's job, on the other hand, is to "pop" a package from some production line. The popped package will then be moved to a target line. The control panel that Ling uses has three buttons for each production line: popping the (1) first / (2) last / (3) highest package from the production line, and moving it to the target line.

Occasionally, some production line may be closed for maintenance. When it happens, all the packages on the closed production line will be dequeued and pushed to another running production line in order. We call this operation to be "merging" the closed production line and the running production line. In other words, the sequence of packages in the closed production line is concatenated to the end of the running production line after the merge operation. The closed production line then becomes empty and will not be used again.

The pushing and merging operations are factory-controlled, and Ling cannot do anything about them. What Ling can do is to decide whether to execute the popping action of the first / last / highest package from one of the production lines after each operation. The decision of the popping executions from Ling forms a particular height sequence of the target line. After Ling's executions every day, he writes down the height sequence of the target line on the daily log of the factory.

Nevertheless, Ling is not the most careful person in the world, and hence sometimes makes mistakes in his writing. An obvious mistake is that the line in the record is not possible from any combination of the popping actions. Ling hopes to capture this kind of mistake before sending the daily log to the factory. Can you help him do that?

Input

The first line contains an integer T, which indicates the number of test cases. The following lines contain T test cases and each test case is formatted as follows:

- The first line of the test case contains three integers N, O, and L. N indicates the number of packages. O indicates the total number of "push" and "merge" operations. L indicates the number of production lines.

- The next O lines are the "push" and "merge" operations, one in each line. The line with `push` comes with two numbers, a package height $1 \le h \le N$ and a production line number $0 \le \ell < L$ to push the package to; the line with `merge` comes with two numbers, a broken production line number $0 \le \ell_b < L$ and a different running production line number $0 \le \ell_r < L$.
- The last line contains the record from Ling, which is a permutation of $\{1, 2, \ldots, N\}$ indicating the heights of the packages on the target line.

Output

If it is possible to arrange the packages to the given order by any combinations of the popping actions, please print `possible` in a single line; otherwise please print `impossible` in a single line.

Constraints

- $1 \le T \le 10$
- $1 \le N, L \le 10^5$
- $N \le O < N + L$

Subtask 1 (15 pts)

- $1 \le N, L \le 1000$

Subtask 2 (25 pts)

- $L = 1$, which also means no "merge" operations

Subtask 3 (10 pts)

- No "merge" operations

Subtask 4 (50 pts)

- No other constraints.

Sample Input 1

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

# of package
# of Query
# of production lines

Sample Output 1

```
possible
possible
```

Sample Input 2

```
2
10 13 5
push 10 3
merge 3 4
push 2 1
push 7 4
push 8 4
push 9 4
push 5 4
merge 1 4
push 4 0
merge 4 2
push 1 2
```

```
push 6 2
push 3 2
10 4 8 7 9 1 3 5 6 2
10 13 5
push 7 1
push 5 1
push 1 1
merge 1 4
push 9 4
push 4 0
push 2 4
push 6 0
push 8 4
merge 4 3
push 3 0
merge 2 3
push 10 3
4 6 9 7 8 3 2 10 5 1
```

Sample Output 2

```
impossible
possible
```

Sample Input 3

```
2
10 19 10
push 3 3
push 1 7
push 10 3
merge 6 2
push 4 8
merge 1 3
push 8 3
merge 9 7
merge 7 3
```

0: 4 6 3

1:

2:

3: 7 5 1 9 2 8 10

4:

```
4 6 9 7 8 3 2 10 5 1
感覺好像拿不出9，不知道是哪裡理解錯誤了？
```

鄭豫澤 (CHENG, YU-CHE)
2021年4月17日

你好，你可以在操作中途任何時間就執行 pop 操作，而不用等到最後才一次 pop 全部，謝謝。
由 鄭豫澤 (CHENG, YU-CHE) 於 4月17日 0:17 編輯

朱俊能 (CHUN NENG CHU)
2021年4月17日

了解，謝謝助教

張芷榕 (CHANG, CHIH-JUNG)
2021年4月17日

順便向助教確認一下，每一次operation之後，是可以選擇pop或不pop，並且pop可以不只一次對嗎？還是最多只能pop一次？謝謝！

林庭風 (Lin Ting Feng)
2021年4月17日

每個operation後都可以選擇若干次pop，並不限於一次～ 簡單來說就是可以把n次pop任意插入原本的操作序列

楊偉倫 (WARRON YIANG WAI LOON)
星期六

hi TA , Is that *push h (1 <= h <= N)* is **distinct?** Or it's may appear twice or above?

Example :

push 2 0

push 2 0

push 3 2

push 2 0

Thanks!

黃品硯 (PIN-YEN HUANG)
星期六

Every h is a distinct number, there will be no number appear more than once.

TA Pin-Yen
由 黃品硯 (PIN-YEN HUANG) 於 4月25日 0:37 編輯

```
push 7 3
merge 8 3
merge 5 3
push 6 3
push 9 3
merge 4 0
push 2 2
merge 3 0
push 5 0
merge 0 2
10 3 6 8 2 7 9 1 5 4
10 19 10
push 7 6
merge 3 2
merge 6 2
push 6 4
merge 7 8
merge 4 5
push 9 5
merge 0 1
push 8 1
push 5 5
push 2 8
push 3 2
merge 8 5
push 1 2
merge 9 1
push 10 2
merge 2 1
push 4 1
merge 1 5
6 9 7 5 3 1 2 10 4 8
```

Sample Output 3

```
impossible
possible
```