2008 ACM ICPC South Central USA Regional Programming Contest

Wandering Aimlessly

Introduction:

Most console role-playing games, such as the famous *Final Fantasy* and *Dragon Quest* games, have towns filled with inconsequential characters that wander around aimlessly, waiting for the player to speak to them.

You've been tasked with the development of the town character (or *NPC*) movement handling for a new game.

Any given NPC has a movement script with a few simple commands:

Command	Behavior
`NORTH x`	Move NORTH (up) `x` steps, one per turn
`SOUTH x`	Move south (down) `x` steps, one per turn
`EAST x`	Move east (right) `x` steps, one per turn
`WEST x`	Move west (left) `x` steps, one per turn
`PAUSE x`	Stay in the current location `x` turns

Note that the individuals who write the NPC movement scripts aren't the most careful coders in the world. Sometimes the scripts have the NPCs attempting to walk through walls and other barriers or even off the edge of the map. When this occurs, all of the steps of movement that would put the NPC in such an invalid location are converted into `PAUSE`s. For example, given this small snippet of a town:

```
...#
.1.#
...#
```

if the NPC represented by the 1 had `EAST 5` as their next command, that would be converted on the fly to `EAST 1` followed by `PAUSE 4`. This processing must be done before determining whether a script is *cyclic* or *reversible*.

A script is *cyclic* if, at the end of the script, the NPC back in their starting position. This sort of script is simply looped indefinitely. The other type of script is *reversible*; if, at the end of the script, the NPC is *not* back in their starting position, they then run a reversed copy of the script, with directions switched (`WEST` becomes `EAST`, `EAST` becomes `WEST`, `NORTH` becomes `SOUTH`, and `SOUTH` becomes `NORTH`) and `PAUSE`s intact. The end result is the character returning to their starting location.

Original script segment	Reversed script segment

`SOUTH 1`
`PAUSE 5`
`EAST 1`
	`WEST 1`
	`PAUSE 5`
	`NORTH 1`

For the sake of this problem, you can assume that no NPCs will ever attempt to occupy the same location on the map at the same time, although one may enter a location on the same turn as a different NPC leaves it, which is valid behavior.

The simulation starts at turn 0; your task is to show, given a map and set of NPCs with their scripts, what the simulation looks like after a large number of turns have passed.

Input:

Input to this problem will begin with a line containing a single integer N ($1 \le N \le 100$) indicating the number of data sets. Each data set consists of the following components:

- A line containing a single integer C ($1 \le C \le 35$) indicating the number of NPCs in this town;
- A line containing two integers H and W ($1 \le H, W \le 40$) representing the height and width of the town;
- H lines representing the town, each with W characters, with:

    - Hash marks (`#`) representing walls and other impassible obstructions;
    - Periods (`.`) representing open spaces; and
    - digits and capital letters representing the starting location of the NPCs, with `1` representing the first one, `2` representing the second (if present), and so on through the digits, then `A` representing the tenth NPC (if present), `B` representing the eleventh (if present), and so on. The space under an NPC's starting location is open.
- *C* sets of lines describing the NPC scripts; each set consists of:
    - A line containing a single character and an integer *L* ($1 < L < 20$), separated by spaces, where the character is one of the digits or letters used on the map to represent a particular NPC, and *L* is the length of that NPC's script; and
    - *L* lines representing the NPC's script, in the format given above, with no value higher than 40.
- A line containing a single integer *T* ($1 \leq T \leq 1000000$) indicating a number of turns to calculate

Output:

For each data set in the output, output the heading "`DATA SET #`*k*" where *k* is 1 for the first data set, 2 for the second, and so on. the next *H* lines, output a representation of the town map, using the same symbols as the input format described above, with the NPCs in the correct locations after the given number of turns have passed.

Sample Input:

```
1
2
5 5
...#.
.1.#.
...#.
...2.
.....
1 4
EAST 5
SOUTH 1
PAUSE 1
WEST 1
2 4
EAST 1
SOUTH 1
WEST 1
NORTH 1
22
```

Sample Output:

```
DATA SET #1
...#.
...#.
..1#.
.....
....2
```