Problem C
Entrapment
Time Limit: 5 Seconds, Memory Limit: 2G

Entrapment is an asymmetric two-player game that is played on a 3×3 square grid. The two players are called the Runner and the Trapper. The grid squares are numbered from 1 to 9 as depicted below:

1	2	3
4	5	6
7	8	9

Before starting the game, the players agree on the number of rounds that the game will last, and on the starting state of the game board. Up to 8 of the grid squares can be marked as *unavailable*. The players also choose who will be the Runner and who will be the Trapper. The Runner then secretly chooses a starting square from among those that are available (i.e., are not marked as unavailable) but does not tell the Trapper their choice.

Each of the game rounds consists of the following steps, in order:

1. The Trapper publicly chooses some subset of the available squares (the empty set is allowed) and asks the Runner, “Are you currently in any of these squares?”
2. The Runner answers truthfully whether or not they are in any of the chosen squares.
3. The Trapper publicly chooses exactly one available square. That square becomes unavailable for the rest of the game. (The Runner might currently reside in that square; if so, nothing special happens.)
4. The Runner secretly moves from their current square to an available orthogonally-adjacent square. If no such square exists, the Runner announces that they are trapped and the Trapper wins the game.

If the Runner has not been trapped by the end of the last round, they prove to the Trapper that they answered all questions truthfully by revealing their choice of starting square and the move that they made during each round. The Runner then wins the game.

Because the Runner’s initial choice of square is secret, as are all of their subsequent moves, the Runner is allowed to “cheat” by not truly committing to a square. At the end of the game, if the Runner can produce a choice of starting square and subsequent moves that do not result in being trapped and are consistent with the answers to the Trapper’s questions during each round, that is enough for the Runner to win the game.

Interaction

This is an interactive problem. Given the number of game rounds and the set of squares that are initially marked unavailable, determine whether the Runner or the Trapper would win assuming optimal play, and then prove it by playing as that role against the judge. The judge will obey all game rules, but may or may not play optimally.

Interaction starts by reading a line of 2 space-separated integers R and U ($1 \le R \le 9$, $0 \le U \le 8$, $R + U \le 9$): the number of rounds in the game and the number of squares that are unavailable at the start of the game.

Next, if $U > 0$, read a line of U space-separated integers s ($1 \le s \le 9$): the labels of the squares that are unavailable at the start of the game. Please refer to the diagram above for how the squares in the grid are labeled. The U labels are guaranteed to be distinct.

Determine whether the Runner or Trapper would win the game with optimal play, given the starting board and number of game rounds. Print a line of output with the string `Runner` if the runner wins with optimal play, and the string `Trapper` otherwise. You will play as that role for the rest of the game; please see the appropriate section below for further instructions on how to interact with the judge in that role.

For the Runner, repeat the following steps R times:

- Read a line of input with a single integer N: the size of the subset of available squares that the Trapper has chosen to ask about. N is guaranteed to be between 0 and the number of available squares left on the board, inclusive.
- If $N > 0$, read a line of N space-separated integers ℓ ($1 \le \ell \le 9$) listing the labels of the squares in the Trapper's chosen subset. The labels are guaranteed to be distinct and all of the chosen squares are guaranteed to be available.
- Print a line of output containing either the string `Yes` or the string `No`. The former informs the trapper that you are currently in one of the chosen squares; the latter informs the trapper that you are not.
- Read a line with a single integer i ($1 \le i \le 9$), the label of the square that the Trapper marks as unavailable. It is guaranteed that square i is a formerly-available square.
- Print a line with the string `Free` to inform the Trapper that you have secretly moved to an orthogonally-adjacent available square and are ready to proceed to the next round. If there are no orthogonally-adjacent squares available, you must print `Trapped` instead and exit; your submission will be judged incorrect for having failed to elude the Trapper.

After you have played R rounds of the game according to the protocol above, print a line with $R + 1$ space-separated integers. The first integer is the label of your chosen starting square; each of the next R integers are the labels of the squares onto which you moved at the end of each of the R rounds. Your moves must be legal and must be consistent with the answers you gave to the Trapper's queries during each round of play. After printing this line, your program must exit.

For the Trapper, repeat the following steps R times:

- Print a line with a single integer N: the size of the subset of available squares that you would like to ask the Runner about.
- If $N > 0$, print a line of N space-separated integers listing the available squares to ask the Runner about. You may list the labels in any order, but the labels must be distinct and must refer to available squares.
- Read a line of input containing a single string: `Yes` if the Runner is in one of your chosen squares, or `No` otherwise.
- Print a line with a single integer i: the square that you are marking unavailable. The label i must be a valid currently-available square.
- Read a line with a single string: `Free` if the Runner has moved to an available square, or `Trapped` if they were unable to do so. After reading the word `Trapped`, you have won the game, and your program must exit. If you read the word `Free` at the end of the Rth round, your program must also exit, though your submission will be judged incorrect since you have failed to trap the Runner.

The judge is guaranteed to answer all questions truthfully.

Notes

Do not forget to flush the output stream after each line that you print and to cleanly exit after the interaction is done. Please also make sure that you follow the above interaction protocol exactly regarding what information to print on which line of output: for example, if the protocol requires you to print a list of space-separated integers on a single line, the judge **will not** accept each integer on its own line.

If the judge receives invalid or unexpected input, it will print -1 and then immediately exit. Your program must detect this error report and cleanly exit in order to receive a Wrong Answer verdict. If your program blocks waiting for further interaction from the judge, or tries to interpret the -1 as a game move, you may receive a different rejected verdict (such as Time Limit Exceeded or Runtime Error) instead of Wrong Answer.

You have been provided with a command-line tool for local testing. The tool has comments at the top to explain its use.

Read **Sample Interaction 1** **Write**

```
Read:
3 6
1 2 3 7 8 9
                        Write:
                        Trapper
                        2
                        4 5
Read:
Yes
                        Write:
                        5
Read:
Free
                        Write:
                        0
Read:
No
                        Write:
                        6
Read:
Trapped
```

Read **Sample Interaction 2** **Write**

```
Read:
2 0
                        Write:
                        Runner
Read:
7
3 1 2 8 9 4 5
                        Write:
                        Yes
Read:
5
                        Write:
                        Free
Read:
4
4 6 7 8
                        Write:
                        Yes
Read:
7
                        Write:
                        Free
                        5 4 1
```