IOI Team Announcement

It is the night after SELN day 2, and the n students at camp are in the common room playing Bartog for the IOI team announcement.

After winning the very first hand (and coming up with a brilliant rule that you are very proud of) you head upstairs to Angus's room. However, when you enter, Angus and Jerry are nowhere to be found![*] Instead, you see a stack of n envelopes, each labelled by student number and containing whether or not that student has made the IOI team.

You start searching the stack of envelopes for your number, before realising that this is a very slow process. Wanting to get the entire team announcement process done as fast as possible, you whip out your trusty laptop and begin coding.

Students come upstairs one at a time and are numbered from 1 to n in the order that they come upstairs. Initially, the *current position* a student is looking at is the top of the stack. We consider the top of the stack to be position 1 and the bottom to be position n.

The student can perform the following actions:

- `N`: Look at the next envelope in the stack, incrementing the *current position*.
- `P`: Look at the previous envelope in the stack, decrementing the *current position*.
- `R`: Look at the top of the stack, resetting the *current position*.
- `M`: Move the envelope at the *current position* to the top of the stack, which also increments the *current position*. Note that performing this action when the *current position* is at the bottom envelope in the stack still increments the *current position* to position $n + 1$ as the student is now looking at the table underneath the stack.
- `T`: Take the envelope at the *current position* and leave, allowing the next student to come upstairs. Note that this effectively resets the *current position* as the next student starts at position 1.

Performing a `N` action at position $n + 1$ or a `P` action at position 1 is not allowed.

You would like to find a strategy for each student to take their corresponding envelope such that the total number of actions that all students make is minimised.

However, since this is a difficult problem to solve, you do not need to find an optimal solution; **instead, points will be awarded based on how many actions your solution uses.**

Subtasks & Constraints

There is only one subtask in this problem. For all testcases, $1 \leq n \leq 1000$.

There is a sliding scale; see the *Scoring* section for details.

Input

- The first line of input contains the integer n.
- The second line contains n integers, the numbers written on the envelopes in the stack from top to bottom.

Output

The first line of output should contain m, the number of actions your strategy takes. The following m lines should each contain one of `N`, `P`, `R`, `M` or `T`, representing an action.

`T` should appear exactly n times in your output, and the final line should be `T`.

[*]You would find out later that they had been in hospital, recovering from lightsaber injuries.

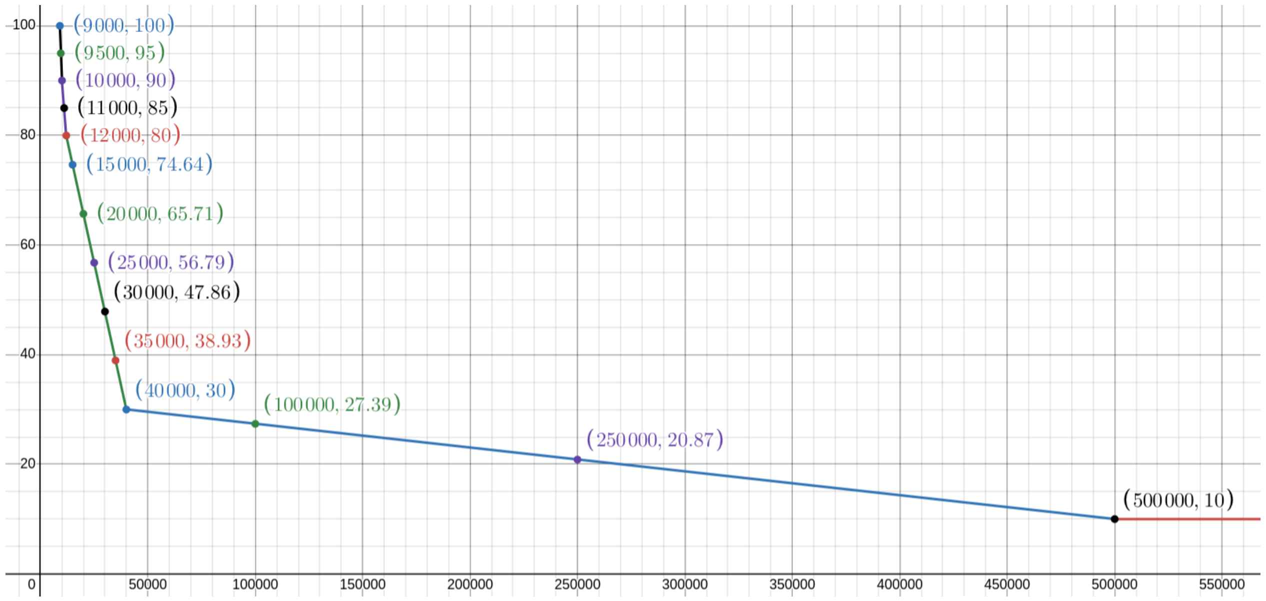
Scoring

If your solution violates any of the conditions in the Output section or does not correctly assign each student their corresponding envelope, then you will score 0.

Otherwise, let M be the maximum number of actions that your solution uses over all testcases.

- If $M \geq 500\,000$, then your score will be 10.
- If $40\,000 \leq M < 500\,000$, then your score will be $30 - 20\left(\frac{M-40000}{500000-40000}\right)$.
- If $12\,000 \leq M < 40\,000$, then your score will be $80 - 50\left(\frac{M-12000}{40000-12000}\right)$.
- If $10\,000 < M < 12\,000$, then your score will be $90 - 10\left(\frac{M-10000}{12000-10000}\right)$.
- If $9000 < M < 10\,000$, then your score will be $100 - 10\left(\frac{M-9000}{10000-9000}\right)$.
- If $M \leq 9000$, then your score will be 100.

The scoring function is depicted in the figure below.



Testing Tool

A testing tool is provided which can generate inputs, test solutions, and compute the scoring function.

To use an existing input file:

```
python testing-tool.py input.txt ./solution
```

To generate a random input file:

```
python testing-tool.py [n] ./solution
```

This will generate an input file called `testing-tool-input.txt` before running your solution.

There is also a `verbose` flag which can be toggled in the source code which enables printing out intermediate states, which may be useful for debugging.

Sample Input

```
4
4 1 3 2
```

Sample Output

```
12
N
T
N
N
M
P
M
R
N
T
T
T
```

Explanation

The following is generated by running the testing tool in verbose mode on the sample input and output:

```
The top of the stack is on the left.
The current position is marked with square brackets.

Initial state:
[4] 1 3 2

Operation: N
4 [1] 3 2

Operation: T
[4] 3 2

Operation: N
4 [3] 2

Operation: N
4 3 [2]

Operation: M
2 4 3 []

Operation: P
2 4 [3]

Operation: M
3 2 4 []

Operation: R
[3] 2 4

Operation: N
3 [2] 4

Operation: T
[3] 4
```

```
Operation: T
[4]

Operation: T
[]

Used 12 operations
Score: 100
```