Reverse (output-only task)

TASK

Consider a Two-Operation Machine (TOM for short) with nine registers, numbered 1…9. Each register stores a non-negative integer in the range 0…1000. The machine has two operations:

S i j	Store one plus the value of register i into register j. Note that i may equal j.
P i	Print the value of register i.

A TOM program includes a set of initial values for the registers and a sequence of operations. Given an integer N ($0 \le N \le 255$), generate a TOM program that prints the decreasing sequence of integers N, N-1, N-2, …, 0. The maximum number of consecutive S-operations should be as small as possible.

Example of a TOM program and its execution for N=2:

Operation	New Register Values									Printed Value
	1	2	3	4	5	6	7	8	9	
Initial values	0	2	0	0	0	0	0	0	0	
P 2	0	2	0	0	0	0	0	0	0	2
S 1 3	0	2	1	0	0	0	0	0	0	
P 3	0	2	1	0	0	0	0	0	0	1
P 1	0	2	1	0	0	0	0	0	0	0

Input cases are numbered 1 through 16 and are available via the contest server.

Input:

- The first line of the input file contains K, an integer specifying the input case number.
- The second line of input contains N.

Example input:

```
1
2
```

Output:

The first line of output should be the string "`FILE reverse K`", where K is the case number.

The second line of output should contain nine space-separated values representing the desired initial values of the registers, in order (register 1, then register 2, etc.).

The rest of the output file should contain the ordered list of operations to be performed, one per line. Thus, the third line contains the first operation to perform, and so on. The last line of the file should be the one that prints 0. Each line should be a valid operation. The instructions should be formatted as in the example output.

Example output #1 (partial points):

```
FILE reverse 1
0 2 0 0 0 0 0 0 0
P 2
S 1 3
P 3
P 1
```

Example output #2 (full points):

```
FILE reverse 1
0 2 1 0 0 0 0 0 0
P 2
P 3
P 1
```

SCORING

Scoring of each test case will be based on correctness and optimality of the TOM program given.

Correctness: 20%

A TOM program is correct if it performs no more than 131 consecutive S-operations and the sequence of values printed by it is correct (contains exactly N+1 integers, starting at N and ending at 0). If any S-operation causes a register to overflow, the TOM program is considered incorrect.

Optimality: 80%

Optimality of a correct TOM program is measured by the maximum number of consecutive S-operations in the program, which should be as small as possible. Scoring will be based on the difference between your TOM program and the best known TOM program.