It's Nut My Job

The Idea

Write a program in the language of your choice that controls a squirrel to relocate acorns in a yard into the fewest number of piles with the least amount of work. The input describes a yard/grid with a squirrel and acorns scattered about.

The squirrel is controlled by only six operation commands:

- **W** : move the squirrel one space to the **West (left)**
- **E** : move the squirrel one space to the **East (right)**
- **N** : move the squirrel one space to the **North (up)**
- **S** : move the squirrel one pile to the **South (down)**
- **P** : **pick up** an acorn from the squirrel's current location if there is one *and* the squirrel is not already holding an acorn
- **D** : **drop** an acorn held by the squirrel (if one exists) onto the squirrel's current location

The Goal

With your squirrel starting at the position dictated by the input with an '@' character, use a minimal number of the operations listed above to reduce the number of piles while earning the **highest** possible score. Each of the six operations listed incurs a cost of one which is counted against your final score. Precisely, your final score is determined as:

$$\left(2 \times AcornCount \times N^3\right) / \left(3 \times PileCount\right) - OperationCount$$ where N is the yard width.

If the squirrel is still holding an acorn at the end, the score is cut in half!

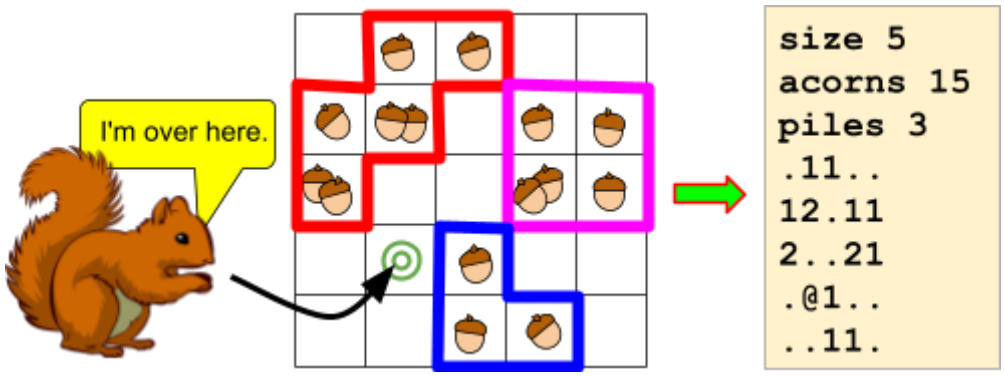
A pile is made up of one or more spaces, each of which must store one or more acorns, sharing either a West/East or North/South edge. In other words, a pile is a bunch of spaces with acorns, connected horizontally and vertically (not diagonally). The example yard from the input shown in the following section contains three piles and has an initial score (no operations applied yet) of 83 ⅓.

Input

Your program should expect input to come into your program as if from a user at the keyboard, you can safely assume no malformed input will be provided.

- The first line indicates the size… how wide and tall the yard is. This is the same as N in The scoring equation.
- The second line indicates how many acorns there are.
- The third line indicates how many piles there are initially in the yard.
- The remaining N lines of input describe the yard's contents, with each line containing N characters of the following breakdown:
    - '@': indicates the initial location of the squirrel
    - '.': indicates there is nothing at that location
    - '0'-'9': indicates how many acorns are at that location

Below is an example yard and the equivalent input your program would receive for it.



Output

The output of your program must be a string of the characters ‘N’, ‘E’, ‘S’, ‘W’, 'P' and 'D' to the console. Please note these letters must be capitalized. All other characters output from your program will be ignored, so if you decide to add output for prompting, debugging or other purposes, just use lowercase letters, numbers and symbols and the testing program will disregard it. Your squirrel cannot move outside the yard; any attempt to will leave your squirrel where it is but cost an extra movement operation anyway.

Evaluation

To evaluate your program, the judges will first compile your program (if your language requires compiling, that is). For the sake of example, let’s assume you submit a source file called prog.cpp. The judges will perform the most basic compile and run your program through our judging script...

```
$ g++ -o prog prog.cpp
$ python3 judge.py nuts.dat ./prog
```

If your program is in an interpreted language, such as Python, it would be run something similar to…

$ python3 judge.py nuts.dat python3 prog.py

This will provide the test data to your program as if it were being entered by someone at the console. It will also capture the output of your program and provide it to the judging program as if it were being entered by someone at the terminal. For your convenience, the judging program's source and data files used for judging are provided.

Runtime

The judges have decided to limit student programs to no more than one minute of runtime.