Задача E. Лабиринт с подсказкой

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Тор похвастался перед карликами, что может пройти любой лабиринт на свете без единой капли волшебства, используя одну лишь лучину. Карлики решили испытать Тора. Они быстрые и искусные строители, и их новый лабиринт обещает быть большим и запутанным. Если слишком долго бродить по нему, лучина угаснет, и карлики посмеются над асом. Посмотрев на строительство, Тор, желая во что бы то ни стало пройти испытание, обратился за помощью к Локи.

Локи мастер на всякие хитрости, так что сможет быстро заполучить карту лабиринта, как только тот будет закончен. Но о том, чтобы передать эту карту Тору, да так, чтобы карлики ничего не заметили, не может быть и речи. Локи сможет передать Тору только коротенькую подсказку...

Помогите Тору и Локи придумать, как будет устроена подсказка, чтобы Тор с её помощью смог пройти лабиринт, пока не угаснет лучина.

Устройство лабиринта

Лабиринт, который строят карлики, можно нарисовать как квадратное поле из $n \times n$ клеток. Между каждыми двумя клетками, соседними по горизонтали или вертикали, либо есть проход, либо стоит стена. Кроме того, весь лабиринт окружён стеной. В левой нижней клетке находится вход в лабиринт, а в правой верхней — выход.

Карлики строят лабиринт так. Они рассматривают все возможные положения стен внутри лабиринта по одному разу в случайном порядке. В каждом таком положении они возводят стену в том случае, если после её появления в лабиринте всё ещё можно дойти из любой клетки в любую другую.

В текстовом виде лабиринт задаётся $2n+1$ строкой, по $2n+1$ символов в каждой. Чётные строки и столбцы, если считать с единицы, соответствуют клеткам, а нечётные — стенам между ними. Символ «`.`» соответствует клетке или проходу, а символ «`#`» — стене или стыку между стенами. В частности, в чётной строке и в чётном столбце всегда будет точка (это клетки), а в нечётной строке и в нечётном столбце всегда будет решётка (это стыки между стенами).

Карту лабиринта из 5×5 клеток, записанную таким образом, можно увидеть в примере ниже. Кроме того, чтобы облегчить локальную проверку, вы можете скачать все лабиринты из тестов с нечётными номерами. Их можно найти по ссылке «Samples ZIP» в интерфейсе тестирующей системы.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза.

При первом запуске решение получает карту и записывает подсказку за Локи. В первой строке записано слово «`view`». Вторая строка содержит целое число n — размер лабиринта ($5 \leqslant n \leqslant 200$). Каждая из следующих $2n + 1$ строк содержит по $2n + 1$ символов. Вместе эти строки задают карту лабиринта.

Вывести нужно одну строку — подсказку, которую Локи передаст Тору. Эта подсказка должна состоять из цифр `0` и `1` и иметь длину от 0 до 1000 символов.

При втором запуске решение получает подсказку и проходит лабиринт за Тора. Этот запуск интерактивный. В первой строке записано слово «`walk`». Вторая строка содержит подсказку, которую Локи передал Тору — ту, что программа вывела при первом запуске. Третья строка содержит целое число n — размер лабиринта, тот же, что и при первом запуске.

Далее решение будет получать кусочек карты, который Тор видит при помощи лучины, и должно в ответ выводить направление его следующего шага. Каждый кусочек карты — это три строки по три символа: клетка лабиринта, где находится Тор, и то, что вокруг неё.

Если решение считает, что Тор прошёл лабиринт и находится на клетке выхода, следует просто корректно завершить работу решения. В противном случае нужно вывести в отдельной строке на-

правление следующего шага: «`N`» для шага на север (вверх по карте), «`W`» для шага на запад (влево по карте), «`S`» для шага на юг (вниз по карте) или «`E`» для шага на восток (вправо по карте). После этого следует очистить буфер вывода: это можно сделать, например, вызовом `fflush (stdout)` в C или C++, `System.out.flush ()` в Java или `sys.stdout.flush ()` в Python.

Если проход свободен, Тор переходит на соседнюю клетку в требуемом направлении и получает очередной кусочек карты, который видит в данный момент. Если же в заданном направлении находится стена, проверка завершается с вердиктом «`Wrong Answer`».

Считается, что решение прошло лабиринт, если оно корректно завершило свою работу, когда Тор находится на клетке выхода, и при этом сделало не более 6000 шагов.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске. В примере мы рассмотрим решение, которое просто записывает в подсказке нужную последовательность шагов: `0` для шага на восток и `1` для шага на север. Конечно, такое решение не всегда будет работать.

Далее показаны два запуска этого решения на первом тесте. При втором запуске добавлены пустые строки, чтобы показать последовательность событий.

стандартный ввод	стандартный вывод
view 5 ########## #.#.......# #.###.##### #.#.....#.# #.#.###.#.# #...#.....# ###.####### #.........# ###.####### #.........# ##########	10001011
walk 10001011 5 ### #.. ### #.# ... ### #.# ... #.# #.# ..# #.# ### #.. #.# #.# ... ### ### ... #.# ### ... ### ### ..# ###	 E N N N E N E E