You wake up and find that you are trapped in a room. Luckily, you have a laptop computer with you, which may help you escape!

You manage to get a map of the room, telling you where you are and where the exit is. You also know that there are two guarding robots inside the room. Normally they just *wander* around the room — they always try to move forward, but if they fail (for example, if they are facing a wall), they turn right until they can go forward. Before a robot moves, if you are in its direct east, south, west or north, and there are no walls separating you from it, it sees you. It then stops wandering, changes its direction and moves towards you. If you get out of its sight, it continues its normal wandering again.

You get caught if you bump into a robot, or if a robot reaches you. **Write a program** that searches for a path leading you to the exit without getting caught.

Input

Input consists of no more than 25 test cases. The first line of each case gives an integer N ($4 \leq N \leq 10$). This is followed by N lines, each with N characters, giving the map of the room.

Key to the map:

- `X`: Your location.
- `#`: A wall; an area that neither you nor the robots can travel to.
- (a space): An area that both you and the robots can travel to.
- `N`: A robot, facing north (upper side of the map).
- `E`: A robot, facing east (right side of the map).
- `S`: A robot, facing south.
- `W`: A robot, facing west.
- `T`: The exit, which is the only place that you can leave the room. Robots see it as a wall; they will not get out of the room.

There are always exactly two robots, and only one exit. The exit must appear on the boundary of the room.

In every step, you move before the robots do. You can either move to the adjacent area in your east, south, west or north, or you can stay where you are. The two robots then move to their adjacent areas in the east, south, west or north, according to the rules specified above. They are thin enough to occupy the same position.

Input ends with a line with only the integer '`0`'.

Output

For each case, your program should print a shortest path (in terms of the number of steps) that leads you to the exit, or the line "`You can't escape...`" otherwise. Use the characters '`N`', '`E`', '`S`', '`W`' and '`X`' to denote moving to the north, east, south, west, or staying where you are respectively. If there are many solutions, print the lexicographically smallest one.

Explanation: The following shows how you escape for the first sample case, step by step. Note how robots change their direction while wandering:

```
########  ########  ########  ########  ########  ########
#S#    #  # #    #  # #    #  # #    #  #N#    #  # #    #
#E     #  #SE    #  #  E   #  #N  E  #  #    E #  #S    E#
# ##   #  # ##   #  #S##   #  # ##   #  # ##   #  # ##   #
###    T  ###    T  ###    T  ###  X T  ###   XT  ###    X
###   ##  ###   ##  ###  X##  ###   ##  ###   ##  ###   ##
##  X ##  ##   X##  ##    ##  ##    ##  ##    ##  ##    ##
########  ########  ########  ########  ########  ########
```

Sample Input

```
8
########
#S#    #
#E     #
# ##   #
###    T
###   ##
##  X ##
########
8
########
#S#    #
#E  #  #
T ##   #
###    #
###   ##
##  X ##
########
0
```

Sample Output

```
ENNEE
You can't escape...
```