Optimus Jr Coding Challenge

Background (fictional)

Optimus Jr is a fictional, pre-prototype robot wandering the factory floor attempting to deliver its cargo. It has only basic sensors and basic abilities to move from its starting location to its destination. Your task, given an input map for each run, is to analyze and plan, if possible, the path that Optimus Jr will take to safely reach its destination. These run results will be key to the evolution of Optimus Jr.

Read the following carefully in order to understand how to interpret the input maps and how to publish your output. We have provided a few sample inputs (maps) and expected outputs. Create additional maps as needed to fully test your logic. Your submission will be tested further against additional maps we have created.

Rules

1. Optimus Jr starts from the **'@'** symbol on the map and is always initially heading SOUTH.
2. Optimus Jr finishes its journey and delivers its cargo when it reaches the location marked **'$'**.
3. Obstacles that Optimus Jr may encounter are represented by **'#'** (unbreakable) or **'X'** (breakable).
4. When Optimus Jr encounters an obstacle, regardless of its current direction, it changes direction using the following priorities: SOUTH, EAST, NORTH and WEST. So, it first tries to go SOUTH; if it cannot, then it will go EAST; if it still cannot, then it will go NORTH; and finally, if it still cannot, then it will go WEST.
5. If Optimus Jr steps on a path modifier, it will cause Optimus Jr to change direction. The **'S'** modifier will make it proceed to the SOUTH, **'E'** to the EAST, **'N'** to the NORTH, and **'W'** to the WEST.
6. Circuit inverters **'I'** reverse the direction priorities that Optimus Jr should choose when encountering an obstacle. Priorities will become WEST, NORTH, EAST, SOUTH. The next inverter Optimus Jr encounters will invert the priorities again, resetting to the original order (SOUTH, EAST, NORTH, WEST), and so on.
7. Optimus Jr may find breaker toggle switches **'B'** along its path that will toggle it in/out of Breaker-mode. Breaker-mode allows Optimus Jr to destroy and immediately pass through breakable obstacles **'X'**. When a breakable obstacle is destroyed, it remains so permanently, and Optimus Jr maintains its course of direction. When Optimus Jr is in Breaker-mode, a breaker toggle switch will toggle it out of Breaker-mode.
8. Teleporters (digits **'1'** through **'9'**) may be present on the map. If Optimus Jr steps on a teleporter, it is immediately teleported to the position of the matching teleporter (the one with the same digit), retaining its direction and mode properties. Teleporters, if present, will always exist in pairs (two **'1'**s, two **'5'**s, etc).
9. Optimus Jr will never start in or be teleported into a single square totally enclosed with unbreakable walls.
10. Finally, the space characters are blank areas on the map having no special effects.

Input (see examples below)

The only input is text consumed from STDIN containing a 2-dimensional grid of characters (the map).
The edges of the map are always unbreakable **'#'** obstacles.
The map always has a starting point **'@'** and a destination **'$'**.
Digits, if present, will always exist in pairs; each pair representing teleporter endpoints (two **'1'**s, two **'5'**s, etc).

Line 1: Two integers, the height **H** and width **W** of the map, separated by a space.
The following H lines: A string of **W** characters representing a horizontal line of the map. The following are the only possible characters (you do not need to check for invalid input): #, X, @, $, S, E, N, W, B, I, 1, 2, 3, 4, 5, 6, 7, 8, 9 and space character.

Output (see examples below)

The only output is text written to STDOUT. It should be the sequence of moves (NORTH, SOUTH, EAST or WEST, 1 per line) taken by Optimus Jr as it travels from its starting point to the destination.
If Optimus Jr cannot reach its destination due to an indefinite loop, then your program must output only LOOP.

Examples

Input

```
5 6
######
#@E $#
# N  #
#X   #
######
```

Expected Output

```
SOUTH
EAST
NORTH
EAST
EAST
```

Input

```
10 10
##########
#        #
#  S   W #
#        #
#  $     #
#        #
#@       #
#        #
#E     N #
##########
```

Expected Output

```
SOUTH
SOUTH
EAST
EAST
EAST
EAST
EAST
EAST
NORTH
NORTH
NORTH
NORTH
NORTH
NORTH
WEST
WEST
WEST
WEST
SOUTH
SOUTH
```

Input

```
10 10
##########
# @      #
# B      #
#XXX     #
# B      #
#    BXX$#
#XXXXXXXX#
#        #
#        #
##########
```

Expected Output

```
SOUTH
SOUTH
SOUTH
SOUTH
EAST
EAST
EAST
EAST
EAST
EAST
```

Examples (cont.)

Input

```
10 10
##########
#    I   #
#        #
#       $#
#       @#
#        #
#       I#
#        #
#        #
##########
```

Expected Output

```
SOUTH
SOUTH
SOUTH
SOUTH
WEST
WEST
WEST
WEST
WEST
WEST
WEST
NORTH
NORTH
NORTH
NORTH
NORTH
NORTH
NORTH
EAST
EAST
EAST
EAST
EAST
EAST
EAST
SOUTH
SOUTH
```

Input

```
10 10
##########
#    1   #
#        #
#        #
#        #
#@       #
#        #
#        #
#    1  $#
##########
```

Expected Output

```
SOUTH
SOUTH
SOUTH
EAST
EAST
EAST
EAST
EAST
EAST
EAST
EAST
SOUTH
SOUTH
SOUTH
SOUTH
SOUTH
SOUTH
SOUTH
```

Input

```
5 5
#####
#   #
# $ #
# @ #
#####
```

Expected Output

```
LOOP
```