Czech ACM Student Chapter

Charles University in Prague
Masaryk University
Pavol Jozef Šafárik University in Košice
Slovak University of Technology

Czech Technical University in Prague

Technical University of Ostrava
University of Žilina
Comenius University
University of West Bohemia

CTU Open Contest 2024 – Practice Session

Etnetera Brevity Challenge

`brevity.(c|cpp|py)`, `Brevity.(java|kt)`

ETNIE is a robot who lives in a maze made of unit squares and surrounded by walls. His A.I. circuits dream of seeing the world outside — some day. Can you help ETNIE to assemble a program that will guide him out through an exit, which can be found in some of the squares?

ETNIE is always positioned in a particular unit square and faces one of the four cardinal directions. He executes a program consisting of instructions, each being represented by exactly one character, so the program can be written as one string. The possible instructions are:

Char	As for	Action
`F`	"forward"	Make one step in the current robot direction, if possible.
`B`	"backward"	Step backwards one square, without changing a direction.
`L`	"left"	Turn 90 degrees to the left.
`R`	"right"	Turn 90 degrees to the right.
`U`	"u-turn"	Turn around 180 degrees.
`(` and `)`		No robot move but marks a nested block of instructions.
`/`	"else"	Skip forward to the corresponding right parenthesis instruction.
`!`	"repeat"	Skip back to the corresponding left parenthesis instruction.
`A`	"ahead"	Condition: Is there a wall immediately in front of the robot?
`S`	"starboard"	Condition: Is there a wall in the square to the right-hand side?
`P`	"port"	Condition: Is there a wall in the square to the left-hand side?

The execution starts with the first character/instruction. After each instruction is executed, the internal program counter is incremented, thus proceeding to the next instruction in the string. After the end of the string is reached, the program terminates and ETNIE stops moving, sighing desperately.

The conditional instructions (`A`, `S`, and `P`) work as follows: If the condition is satisfied (there *is* a wall in the respective direction), the program execution skips forward to the next "`)`" or "`/`" in the current nesting level. The following examples show a typical and intended usage of conditions but feel free to invent your own enhancements.

Instructions	Meaning
`(PL)`	**if** (*not* wall to the left) { turn left; }
`(AF!)`	**while** (*not* wall ahead) { make 1 step; }
`(SRF/L)`	**if** (*not* wall to the right) { turn right; make 1 step; } else { turn left; }

Whenever a nested block is encountered while skipping forward or backward in search for a matching left or right parenthesis (or slash), the inner block is skipped as a whole. If there is no matching left or right parenthesis, the execution skips to the start or the end of the program, respectively. If ETNIE is instructed to make a step into a wall, he refuses to do so and does not move. He also remains calm when an unknown character is encountered in the program.

Input Specification

The input contains 42 test cases. The first line of each case specifies the maze size by two integers R and C ($3 \leq R, C \leq 100$). The next R lines contain C characters each. The possible characters are simple spaces (" ", ascii 32) for free squares, "hash" signs ("`#`", ascii 35) for walls, uppercase "`E`" for exits, and "`<`", "`>`", "`^`", or "`v`" (ascii 60, 62, 94, or 118) for the initial robot position and orientation. There is always exactly one of these four characters in each maze. All characters at the maze border will be either walls or exits.

Output Specification

You must print exactly one line of output for each test case, containing any program that will *eventually* guide ETNIE to an exit in the given maze. It is not necessary for the program to terminate at the exit square.

Your program must not be longer than 10 000 characters. The number of steps made by ETNIE is irrelevant and generally not limited, as he has enough time to pursue his lifetime dream. It is guaranteed that such a program exists.

Sample Input *(for practical reasons, this Sample Input does not include all 42 test cases)*

```
3 42
##########################################
#<                                       #
##############E###########################
8 60
############################################################
# #                        #          #                   #
#                          #        #              #     ##
# #               E                                       #
# #                                    #                 #
# ############################                            #
#                      ^#EEEE#                            #
E########################EEEE#####################################
```

Output for Sample Input

```
LLFFFFFFFFFFFFLLLF
RFP/(FA!)!)L((AF!)(PL/R)!
```

While any valid output of this problem will be accepted for the purposes of the Practice Session and its ranking, our **Etnetera Brevity Challenge** seeks for the *shortest* solution. All *correct* runs will be ranked by the sum of the lengths of all 42 programs produced, and the team with the shortest output will be awarded a real robot!
(For practical reasons, this price is only applicable for Prague teams.)

You may submit your solutions repeatedly — the best result counts. Good Luck!