Czech ACM Student Chapter
CTU in Prague, Dept. of Computer Science and Engineering
Charles University in Prague, MFF
VŠB TU Ostrava, FEI, Dept. of Computer Science
SUT Bratislava, FEI, Dept. of Computer Science and Engineering

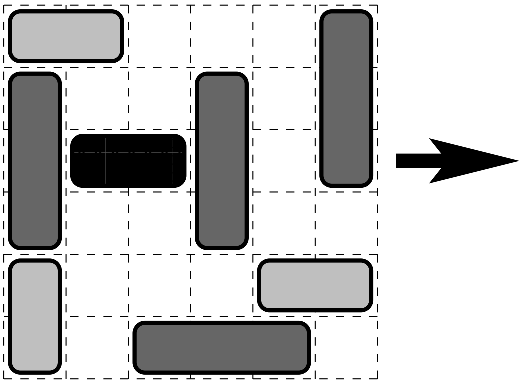
CTU Open Contest 2003

Traffic Jam

`traffic.p`, `traffic.c`, `traffic.C`

Traffic jam is a real nightmare of all drivers. Nobody likes to be stuck in the overfilled streets, when the cars move very slowly, if they even move at all. Professional drivers face traffic jams quite often. Truck drivers of ACM are not an exception. Can you help drivers to find the way out of the traffic jam?

We can model a small (but complicated) traffic jam on a 6×6 grid of squares. Vehicles (cars and trucks) are scattered over the grid at integer locations, as shown below. Both types of vehicles are 1 square wide. Cars are 2 squares long, and trucks are 3 squares long. Vehicles may be oriented either horizontally (East-West) or vertically (North-South) relative to the grid.



Vehicles cannot move through each other, cannot turn, and cannot move over the edge of the grid. They can move in their direction (horizontally-oriented vehicles cannot move vertically and vice versa), as long as they are not blocked by another vehicle or by the edge of the grid. Only one vehicle may move in a single step, but it may move by as many squares at a time as possible, providing there is enough empty space.

Our goal is to move vehicles back and forth until a particular horizontally-oriented vehicle (your own car — the black one on the picture above) leaves the rightmost (eastern-most) edge of the grid, where it is considered to have escaped the traffic jam. You are to write a program that will find a solution requiring the minimum possible number of moves.

Input Specification

The input file will consist of one or more input scenarios. Each scenario begins with a single integer n, $1 \leq n \leq 10$, giving the number of vehicles in the scenario. Then there will be 6 lines of input, each containing 6 characters. Each character is either a dot ("`.`") representing an empty square, or a lowercase letter representing a vehicle. Your own vehicle is always oriented horizontally and represented by "`x`" characters. The other vehicles use the letters sequentially, beginning with "`a`".

The last scenario will be followed by a line containing a single zero.

Output Specification

For each scenario, output a single line with the statement "`Scenario #`K `requires` X `moves.`", where K is the number of the scenario (starting with 1) and X is the minimum number of moves required to escape the the traffic jam with the particular car.

If it is not possible to escape, output the sentence "`You are trapped in scenario #`K`.`" instead.

Sample Input

```
8
aa...b
c..d.b
cxxd.b
c..d..
e...ff
e.ggg.
8
abbc..
a..c..
axxc..
..gddd
..g..e
..fffe
0
```

Output for Sample Input

```
Scenario #1 requires 8 moves.
Scenario #2 requires 25 moves.
```