Elevator Problem Description

The goal is to write a program which satisfies the following spec. Implement in any language. Only use standard libraries. The solution should compile and run on a box that includes a vanilla version of the relevant compiler/interpreter. Please include build instructions with your solution.

Priorities:

1.Correctness
2.Elegance and clarity of code

The following diagram is an elevator state – the state of an elevator system at a given point in time.If the diagrams do not form a rectangle, cut and paste into a document with a monospace font like Courier.

Dots represent an elevator shaft.

Letters (specifically uppercase A-Z) represent an elevator – for example, elevator A is on the 1st floor (1-indexed), and the following elevator state has 5 floors:

Note that elevator shafts can abut exterior walls, and elevator shafts can rise to the top floor, as in the following example:

Input: Take a series of elevator states from a file, representing successive states of the elevator system from t = {1,2,3...} Each elevator state will be separated by a single empty line. The first and last lines of the file will not be empty. Valid elevators are specified by the uppercase characters A-Z. Lines may be broken by \n or by \r\n.

```
xxxxxxxxxxxx
xx.x.x.x.xxx
xx.x.x.x.xxx
xx.xBx.x.xxx
xx.x.xCx.xxx
xxAx.x.xDxxx
```

```
x.xBx.x.x
```

```
x.x.xCx.x
xAx.x.xDx
```

An elevator system with elevator states for t=1, 2, and 3 might look like:

```
xxxxxxxxxxxx
xx.x.x.x.xxx
xx.x.x.x.xxx
xx.xBx.x.xxx
xx.x.xCx.xxx
xxAx.x.xDxxx
xxxxxxxxxxxx
xx.x.x.x.xxx
xx.xBx.x.xxx
xx.x.x.x.xxx
xxAx.x.x.xxx
xx.x.xCxDxxx
xxxxxxxxxxxx
xxAx.x.x.xxx
xx.x.x.xDxxx
xx.xBx.x.xxx
xx.x.xCx.xxx
xx.x.x.x.xxx
```

Write a command line program. It should take exactly three args as specified:

<program invocation> <elevator system filename> <starting elevator> <final destination>

Where <final destination> is specified as <floor>-<time>, e.g. 3-2, indicating that the final destination is the 3rd floor at time t=2. At t=1, you begin in the elevator specified by <starting elevator>, e.g. “A”.

<program invocation> can be whatever string is necessary to invoke the program.

In a given time period, you can board an elevator or stay on the elevator you are on. In any time period, you can only board any elevator on the same floor as the elevator that you are currently on (including t=1).

The goal is to find a series of legal actions that lead to you being at the final destination – the right floor at the right time.

The set of actions should be written to stdout as a series of actions in a single string. An action is defined as the elevator you board (or stay on) at time t, eg:

AABDD

Indicates that you were on (or switched to) A at t=1, stayed on A at t=2, switched to B at t=3, and switched to D at t=4, and stayed on D at t=5. If the final destination time is T, your actions should have T-1 actions in it (the elevator you are on at time T-1 must transition to the correct floor at time T for your solution to be valid).

If there is a solution, the solution string is the *only* thing that should be printed to stdout.

If there is no solution, print ”No Solution” to stderr, and *nothing* to stdout.

It is important that the output instructions be followed exactly, as we evaluate the correctness of your program using a test harness.

Your program may assume that the elevator system file conforms to the specification, and that the arguments to the program are valid and in range of the elevator system.