```
Stage 3
=======
maze has a solution with cost 10
##00##############
##++02++04++06++##
########++########
##++08++06##----##
##10########--####
```

Further examples showing the full output that is required are provided on the LMS, and you should study them carefully so that you understand the details of what is required in this stage. Note that the output from this stage is *in addition* to the output of Stages 1 and 2.

Stage 4 – Plotting a Path (marks up to 20/20)

Now add further state information to the struct for each cell so that the exact path implied by one solution is drawn, and none of the other cell costs are shown. In this stage, reachable cells not on the final path should be shown as doubled blanks. The required output from this stage for `test0.txt` is

```
Stage 4
=======
maze solution
##00##############
##..02..04      ##
########..########
##..08..06##----##
##10########--####
```

In cases where there are two or more exit gaps that have the same minimum distance from a start gap, a path to the leftmost of them should be plotted. Note that the output from this stage is *in addition* to the output of Stages 1, 2 and 3, assuming that a solution exists. In cases where there is no solution nothing should be printed except for the Stage 1, 2 and 3 output. Further examples showing the full output that is required are provided on the LMS, and you should study them carefully so that you understand the details of what is required in this stage.

A Note on Algorithms

You are free to adopt any approach that you wish to labeling cells and computing path costs, but you do need to be systematic, and develop a mechanism that computes the correct answers. Be sure to provide comments in your programs to help the markers understand the particular mechanism you have used.

One possible approach is to cycle through the maze, examining every cell in order. Then, if that cell has been labeled with a path cost, use that cell to try and also label its neighbors with a path cost that is one greater. Path costs of labeled cells should only ever decrease, once a cell is first labeled. If a complete run through of every cell results in no changes to the path cost of reaching any cell in the maze, then a final set of path costs must have emerged. On the other hand, if any cell got its cost reduced in the last run through, start another pass through and allow that change to propagate further if it needs to. Begin by assigning a path cost of zero to the gaps in the top row of the maze.

This isn't a very *efficient* algorithm, but it will be fast enough for the scale of maze being considered here. There are – of course! – more efficient algorithms than this that can be applied when there are millions or even billions of cells involved (for example, when the maze is three-dimensional). Come back and enrol in comp20003 in second semester if you want to know more.