Problem 2:

First 12 values:

row: 3

col: 5

\*\*\*\*

row: 3

col: 6

\*\*\*\*

row: 3

col: 4

\*\*\*\*

row: 3

col: 3

\*\*\*\*

row: 2

col: 4

\*\*\*\*

row: 1

col: 4

\*\*\*\*

row: 1

col: 3

\*\*\*\*

row: 1

col: 2

\*\*\*\*

row: 1

col: 1

\*\*\*\*

row: 2

col: 1

\*\*\*\*

row: 4

col: 5

\*\*\*\*

row: 5

col: 5

Problem 4:

First 12 values:

row: 3

col: 5

\*\*\*\*

row: 4

col: 5

\*\*\*\*

row: 3

col: 4

\*\*\*\*

row: 3

col: 6

\*\*\*\*

row: 5

col: 5

\*\*\*\*

row: 2

col: 4

\*\*\*\*

row: 3

col: 3

\*\*\*\*

row: 6

col: 5

\*\*\*\*

row: 5

col: 4

\*\*\*\*

row: 1

col: 4

\*\*\*\*

row: 7

col: 5

\*\*\*\*

row: 5

col: 3

The difference between the stack and the queue algorithm is the way in which each algorithm searches the maze for a path. Stacks uses depth-first search while queues use breadth-first search. Depth-first search starts at the starting position and explores each path all the way down until a dead-end is reached. Depth-first search basically continues down a path until it can’t go further anymore. And then the algorithm resumes at the last place it diverged from. Breadth-first search starts at the starting position and explores all neighboring spaces first before moving onto the next path with new neighbors. The correct path is determined one step at a time instead of tracing all the possible paths to see which ones are wrong and which one is the right one. Breadth-first searching is a lot more efficient.