Homework 2, by Tomas Kaljevic

* 1. (6, 4)
  2. (6, 3)
  3. (6, 5)
  4. (7, 5)
  5. (8, 5)
  6. (8, 6)
  7. (8, 7)
  8. (8, 8)
  9. (7, 8)
  10. (6, 6)
  11. (5, 4)
  12. (4, 4)
  13. (6, 4)
  14. (5, 4)
  15. (6, 5)
  16. (6, 3)
  17. (4, 4)
  18. (6, 6)
  19. (7, 5)
  20. (3, 4)
  21. (4, 5)
  22. (8, 5)
  23. (2, 4)
  24. (4, 6)

In the function that uses a stack to see if there exists a path to the end of the maze, the stack can only pop the most recently added coordinate pair (i.e. the top of the stack). Thus, the function follows the path given by that coordinate pair until that path becomes a dead end or reaches the end coordinate pair. For example, in my function, if there is a valid coordinate pair west of the starting position, the function will traverse the path given by that coordinate pair until it hits a dead end or completes the maze. Conversely, the function that uses a queue can only dequeue the coordinate pair at the front of the queue (i.e. the first one added to the queue). Thus, the function searches every possible coordinate pair that is a neighbor to the starting position (or the position that was dequeued most recently) until there are no more coordinate pairs to search or the end coordinate pair is reached. For example, in my function, if there is a valid coordinate pair one north of the starting position, the function will check each of the neighbors of that north coordinate pair to see if it’s the end of the maze. If not, it will then check the neighbors of the coordinate pair one east of the starting position (if it exists), and so forth.