**QUESTION 2**

The first 12 (r,c) coordinates popped from the stack:

1. First pops (sr, sc) = (3,5)

**Ran the function and printed the coords that were being popped:**

**TO CHECK MY WORK:**

**3,5**

**3,6**

**3,4**

**2,4**

**1,4**

**1,3**

**1,2**

**1,1**

**2,1**

**3,3**

**4,5**

**5,5**

**5,4**

**5,3**

**6,3**

**6,2**

**6,1**

**5,1**

**4,1**

**6,5**

**7,5**

**8,5**

**8,6**

**8,7**

**8,8**

* 1. STACK

(3,6)

(3,4)

(4,5)

1. Pops (3,6)
   1. STACK

(3,4)

(4,5)

1. Pops (3,4)
   1. STACK

(2,4)

(3,3)

(4,5)

1. Pops (2,4)
   1. STACK

(1,4)

(3,3)

(4,5)

1. Pops (1,4)
   1. STACK

(1,3)

(3,3)

(4,5)

1. Pops (1,3)
   1. STACK

(1,2)

(3,3)

(4,5)

1. Pops (1,2)
   1. STACK

(1,1)

(3,3)

(4,5)

1. Pops (1,1)
   1. STACK

(2,1)

(3,3)

(4,5)

1. Pops (2,1)
   1. STACK

(3,3)

(4,5)

1. Pops (3,3)
   1. STACK

(4,5)

1. Pop (4,5)
   1. STACK

(5,5)

1. Pop (5,5)
   1. STACK

(5,4)

(6,5)

Continued…

As we can see, we don’t ever reach our end coordinate.

**PROBLEM 4**

**Review this on paper**

The first 12 (r,c) coordinates popped off of the queue:

**Ran the function and printed the coords that were being popped:**

**TO CHECK MY WORK**

**3,5**

**4,5**

**3,4**

**3,6**

**5,5**

**3,3**

**2,4**

**6,5**

**5,4**

**1,4**

**7,5**

**5,3**

**1,3**

**8,5**

**6,3**

**1,2**

**8,4**

**8,6**

**6,2**

**1,1**

**8,3**

**8,7**

**6,1**

**2,1**

**8,2**

**7,7**

**8,8**

1. Pops (sr,sc) = (3,5)

QUEUE: (4,5) (3,4)

1. Pops (4,5)

QUEUE: (3,4) (3,6) (5,5)

1. Pops (3,4)

QUEUE: (3,6) (5,5) (3,3) (2,4)

1. Pops (3,6)

QUEUE: (5,5) (3,3) (2,4)

1. Pops (5,5)

QUEUE: (3,3) (2,4) (6,5) (5,4)

1. Pops (3,3)

QUEUE: (2,4) (6,5) (5,4)

1. Pops (2,4)

QUEUE: (6,5) (5,4) (1,4)

1. Pops (6,5)

QUEUE: (5,4) (1,4) (7,5)

1. Pops (5,4)

QUEUE: (1,4) (7,5) (5,3)

1. Pops (1,4)

QUEUE: (7,5) (5,3) (1,3)

1. Pops (7,5)

QUEUE: (5,3) (1,3) (8,5)

1. Pops (5,3)

QUEUE: (1,3) (8,5) (6,3)

Differences between the two algorithms:

The algorithm that uses the stack is “depth-first”, in that it will follow one path all the way until it is cornered before retracing its steps to the next possible path. This occurs as a result of the stack being a LIFO data structure. The last element in (which is the farthest from the starting node) will be the first one popped; after the spots next to that element are scanned, they are added to the top of the stack, and one of these elements is popped, continuing the trend of popping and scanning elements farthest down one path from the start point.

The algorithm using the queue is “breadth-first”, in that it will scan all elements a distance *n* from the starting node prior to scanneing elements a distance of *n+1* away. This occurs as a result of the queue being a FIFO data structure. When we add a particular coordinate to our queue, we know that it will be scanned in the order of addition; this means that we will add all elements *n* elements down all paths in the same iteration of the loop, and visit them in that order.