2. Given the algorithm, main function, and maze shown at the end of problem 1, what are the first 12 (r,c) coordinates popped off the stack by the algorithm?

The first 12 (r.c) coordinates are

(6,4), (6,3), (6,5), (7,5), (8,5), (8,6),

(8,7), (8,8), (7,8), (6,6), (5,4), (4,4).

4. Given the same main function and maze as are shown at the end of problem 1, what are the first 12 (r,c) coordinates popped from the queue in your queue-based algorithm?

The first 12 (r,c) coordinates are

(6,4), (5,4), (6,5), (6,3), (4,4), (6,6)

(7,5), (3,4), (4,5), (8,5), (2,4), (4,6)

How do the two algorithms differ from each other? (Hint: how and why do they visit cells in the maze in a different order?)

The stack algorithm first visits the starting cell and pushes the cells near it into the stack. Then it tries to access the cell near the starting cell in each direction, in the order of West, South, East and North, because the cells are pushed into the stack in the opposite order and stack is “First in, last out.” If one of the cells in these directions is not visited and is not a wall, the program visits that cell and pushes the cells near it into the stack. The program will then try to visit the cell near this cell in each direction, again in the order of West, South, East and North. It will repeat the process until all the possible paths after the second cell are visited and then tries to access the cell near the starting point in the next direction.

The queue algorithm also visits the starting cell first and pushes the cells near it into the stack. Then it tries to access these cells in the order of them pushed into the stack. It will visit all the cells next to the starting cell before it proceeds to the cells that are pushed after Because queue is “first in, first out.” The cells around the starting cell is first pushed, they will be the first popped out the stack. The algorithm will expand its searching radius each time until it finds the ending cell.