Joshua Rampersaud
Fall 2020; Professor Fried
Question 2, 3, 7

Explanation for Problem 2:

When talking about prefix stack or even postfix stacks, these definitely works faster than the infix stack. In postfix and prefix expressions which ever operator comes before will be evaluated first, irrespective of its priority. Also, there are no brackets in these expressions. We can convert infix to postfix and can convert infix to prefix. First, I Put a pointer P at the end of the end. If character at P is an operand push it to Stack. If the character at P is an operator pop two elements from the Stack. Operate on these elements according to the operator, and push the result back to the Stack
Decrement P by 1. As long as there are characters left to be scanned in the expression. The Result is stored at the top of the Stack.

Explanation for Problem 3:

When it comes to testing a stacks speed, using a linked list based stack, you would Implement a stack using single linked list concept. Linked list means what we are storing the information in the form of nodes and we need to follow the stack rules and we need to implement using single linked list nodes. A simple rule that is last in first out and all the operations we should perform so with the help of a top variable only with the help of top variables. The main advantage of when using linked list over an array's is that it is possible to implements a stack that can shrink or grow as much as needed. In using array will put a restriction to the maximum capacity of the array which can lead to stack overflow.

Explanation for Problem 7:

When writing this code in order to solve a maze, This code should be able to print a path of straight 'x's from start ([1][1]) to end ([8][13]). As it is running, it will be working by checking for an empty spot in the order of North, South, East and West. If there are no empty spots, then it marks the spot as a dead end (d) and backtracks by popping the stack.