Problem Set 2

- Create stack and queue to be used as lists. They differ in their LIFO or FIFO retrieve methods. We should be able to

    1. add items with .add method
    2. retrieve items with .retrieve method
    3. count the number of items in the stack or queue with the .length method.

- Update the Martingale game in a way that regardless of the number of outcomes in a gamble, you obtain a profit of (outcomes-1) whenever you win a round. This means, you start with $1, so that if you win in the first round you get outcomes back, meaning outcomes-1 profit. If you lose, you keep playing that critical amount in each round so that a set of rounds always gives the same profit of outcomes-1 unless you lose until you lose all your fortune. Then do the same with a 5 outcome game that pays back 4 whenever your win. Next, compare the two outcome and five outcome games in a setting where you make (outcomes+1) whenever you win a round. Show experimentally that in a 10,000 round game, your chances of winning are larger when the payback amount is more. You should compute this report on the console.

- We applied the greedy algorithm based on value density. Do the same for max value and min weight. How do they perform? Using the optimal solution from Gurobi, compute 3 approaches objective performances.

- Randomly generate Knapsack problems of certain size. Use random seed number to ensure your problem instances can be replicated. Next, iteratively increase the size of the problem (number of items) and report what happens when there are 10, 20, 50, 100, 200, 500, 1000, 2000, 5000 items? Is there a difference in how solution time progresses with increase problem size?

- In the Knapsack problem, write a solution algorithm that randomly adds items to the knapsack, as long as they fit. Use random seed number to ensure your algorithm and results can be replicated. How would that greedy algorithm compare to the random selection algorithm in terms of time and objective value?