# Assignments

1. Consider any of the following real life situation. Formulate a question and then design a simulation that can help to answer it. Which linear data structure would be apt to solve the situation?

Possible situations include:

* + Cars lined up at a car wash
  + Customers at a grocery store check-out
  + Airplanes taking off and landing on a runway
  + A bank teller

Be sure to state any assumptions that you make and provide any probabilistic data that must be considered as part of the scenario.

1. Implement a direct infix evaluator that combines the functionality of infix-to-postfix conversion and the postfix evaluation algorithm. Your evaluator should process infix tokens from left to right and use **two stacks**, one for **operators** and one for **operands**, to perform the evaluation.
2. Design and implement a class that will compare the performance of a C++ vector with a list implemented as a linked list.
3. In the hash table map implementation, the hash table size was chosen to be 101. If the table gets full, this needs to be increased. **Re-implement** the **put** method so that the table will automatically resize itself when the loading factor reaches a predetermined value (you can decide the value based on your assessment of load versus performance).
4. This problem is called the string edit distance problem, and is quite useful in many areas of research. Suppose that you want to transform the word “**algorithm**” into the word “˘.” For each letter you can either copy the letter from one word to another at a cost of 5, you can delete a letter at cost of 20, or insert a letter at a cost of 20. The total cost to transform one word into another is used by spell check programs to provide suggestions for words that are close to one another. Use dynamic programming techniques to develop an algorithm that gives you the smallest edit distance between any two words.
5. Modify the code for a binary search tree to make it threaded. Write a non-recursive **inorder traversal method** for the threaded binary search tree. A threaded binary tree maintains a reference from each node to its successor.
6. Design and implement class to solve word ladder problem using graphs. Consider the following puzzle called a word ladder. Transform the word “**FOOL**” into the word “**SAGE**”.

Considerations: In a word ladder puzzle you must make the change occur gradually by changing **one letter at a time**.

At each step you must **transform one word into another word**, you are **not allowed to transform a word into a non-word**.

The word ladder puzzle was invented in 1878 by Lewis Carroll, the author of *Alice in Wonderland*. The following sequence of words shows one possible solution to the problem posed above.

FOOL

POOL

POLL

POLE

PALE

SALE

SAGE

Represent the relationships between the words as a graph.

Use the graph algorithm (breadth first search) to find an efficient path from the starting word to the ending word.

1. Design a system that cycles through traffic light signals (red, yellow, green) using a circular queue data structure to simulate continuous rotation of the signals.
2. Create a program to match passengers with available taxis based on proximity and waiting time using either min-heap or priority queue
3. Develop a system that manages the landing and takeoff of flights. The flights are prioritized based on their fuel levels.
4. Build a warehouse inventory system where products can be searched, added, and removed quickly using a Hash Map
5. Implement a system that tracks price changes of products over time and returns the maximum, minimum, or average price efficiently using Binary Search Tree
6. Design a system that models a user’s friend network where mutual friendships can be identified using Graph
7. Design a system to track packages as they move through multiple delivery points until reaching the final destination using Graph
8. Implement a matchmaking system that pairs players based on their skill levels using a balanced binary tree
9. Implement a system that logs all banking transactions, allowing efficient access to the last 10 transactions using a linked list
10. Create a leaderboard for a game where scores are updated in real-time, and the top players can be displayed.
11. Implement a system using graphs that recommends products to users based on their past purchases using similarity in product categories
12. Implement a stock buy-sell system where you can efficiently track the highest and lowest prices using stack
13. Implement a system that tracks trending topics based on hashtags over time using Trie