Please note: I am sure you will have no trouble finding solutions to these online if that is what you choose to do. I would like to discourage you, in the strongest possible terms, from doing so. The goal of these is to help you get into the mindset of solving the sorts of problems you will solve in 260P. Similarly, if you have seen this exact problem before, do not try to remember the solution – rather, try to solve it or, if you happen to remember the solution, to walk through the process of solving it.

This is a warm-up/review problem set covering topics from prerequisites you will need for CompSci 260P. The purpose of this problem set is for you to establish how well prepared for the course you are. While you might not be able to answer each of these offhand, you should be able to with a little bit of work and review. If you have trouble with these problems, that should indicate to you that you have some prerequisite material to review.

This is not a practice test. If I were to assign and collect a problem set 0 rather than providing this, it would probably be 4-5 problems similar to the ones here.

How to use this document: Read through the questions so you can get an idea for what sorts of things you would ideally be able to answer. Note that I am not asking you to answer them in any very constrained period of time – for that matter, some of these are of too high a difficulty level to give as a question during a test of prerequisites. Rather, choose a few problems, attempt them, and then discuss them with your study group, who will ideally have already attempted them as well. Note that looking up an answer, or me providing one for you, would defeat the point of this exercise. You should never attempt to look up the solution to something in this class beyond what is provided by the professor during the quarter. Then, repeat with other questions, until you have completed the set.

You should think about whether you were able to start a problem effectively, whether the terms used made sense to you, and whether you would be able to explain your technique to a friend in your study group who has also attempted the problem. For most of these exercises, the best feedback is not whether or not you solved it correctly, but the ease or lack thereof with which you were able to attempt them.

Please note that this is not strictly from one-level of prerequisite. Some of these questions would be review questions for the prerequisite as well. Some topics are needed across many classes!

Some of these questions come from textbooks that are used in prerequisite classes. For clarity of exposition, these are not explicitly cited.
1 Topics

1. Recursion

2. Asymptotic Notation ($\mathcal{O}, \Omega, \Theta$)

3. Fundamental Graph Algorithms
   (a) Breadth-First and Depth-First Search
   (b) Topological Sort
   (c) Dijkstra’s (Single-Souce, Shortest Path) Algorithm

4. Mathematical Proofs
   (a) Direct, by contra-positive, and by contradiction
   (b) Induction

5. Stacks and Queues
   - Conceptually, although being able to implement one or both would be valuable.
   - More importantly, being able to use one and recognize when one or the other would be appropriate to use.

6. QuickSort and MergeSort algorithms

7. Usage of Stacks and Queues

2 Problems

1. There are $n$ pirates a particular ship. Each has a number, 1…$n$, where a higher number indicates a higher rank; the captain is ranked $n$. When the pirates capture a collection of gold, the captain will propose to the crew a possible manner to split the coins. The splits must be integer: you cannot give someone half a coin. Each member of the crew, including the captain, gets to vote “yes” or “no” on this. If half or more of the crew vote “yes,” then the gold is split this way. Otherwise, the captain walks the plank (is removed from the ship), the previous pirate marked $n-1$ is now captain, and the process continues. For purposes of this problem, each pirate wants to remain on the ship (first priority) and get the most gold possible (second priority).

   Suppose you are the captain of a crew that has $n = 5$. You have captured 100 gold coins. What split is best (for you) to propose? Why?
2. Multiple students have reported having been asked a version of the following question in an
interview. I liked it so much I wanted to put it on this problem set.

Suppose you have a vector $V$ of $n$ distinct numbers. The vector is sorted least to greatest.
You want to create a vector $V'$, such that each value in $V$ has its square somewhere in $V'$. If
$x$ and $-x$ are both in $V$, it is okay to have $x^2$ once or twice in $V'$. Regardless, you want $V'$
to be in sorted order.

(a) Give an algorithm to find $V'$ in $O(n \log n)$ time.

(b) Can you find an algorithm with running time $O(n)$?

(c) Can you find an algorithm with running time better than $O(n)$? Why or why not?

3. A number maze is an $n \times n$ grid of positive integers. A token starts in the upper left corner;
your goal is to move the token to the lower-right corner. On each turn, you are allowed to
move the token up, down, left, or right; the distance you may move the token is determined
by the number on its current square. For example, if the token is on a square labeled 3, then
you may move the token three steps up, three steps down, three steps left, or three steps
right. However, you are never allowed to move the token off the edge of the board. Your
goal is to find the minimum number of moves required to solve a given number maze, or to
correctly report that the maze has no solution.

For example, in the following maze, we can solve this with eight moves: right, down, left,
right, up, left, right, down.

Describe how to use a graph to solve this problem. A complete answer includes a description
of how to form a graph from an arbitrary number maze as well as how to find a solution to
the number maze using the graph, or to report that none is possible.

```
3 5 7 4 6
5 3 1 5 3
2 8 3 1 4
4 5 7 2 3
3 1 3 2
```

4. Suppose I want to have a “Blue and Gold Stack” class. This is a class that has color-coded
push, pop, top and size functions; for example, there is a “blue push” and a “gold push.”
Explain how you can implement this using a single array whose capacity is set at some value
$C$, which will always be larger than the combined sizes of the blue and gold stacks. Your
approach may only use the array of size $C$ and $O(1)$ additional space. You do not need to
write code, but instead describe how you would implement the eight functions.

You will earn no credit if you solve this by simply having two traditional stacks as private
data members and redirecting the calls to $\text{bluePush()}$ to the blue one, etc.
5. Consider the following problem that humanity might face when we make first contact with an alien race. We know that the aliens’ written language operates in a manner similar to English; their alphabet contains $n$ characters, which we will denote $c_1, c_2, \ldots c_n$. We also know that they have a concept of alphabetical order, also similar to English, where some character (not necessarily $c_1$) is first (like ‘A’ in English), some other character is second, and so on. Comparison of two words for alphabetical order works the same way in their language that it does in English.

We are given a list of $m$ words in the aliens’ language, written in an order that follows their alphabetical order. Our goal is to determine a feasible alphabetical order for the aliens’ alphabet that is consistent with the word list given.

Give an efficient algorithm to solve this problem and analyze its runtime. Your algorithm should output the feasible alphabetical order for the aliens’ alphabet if one is consistent with the input; alternately, if we have evidence that the aliens lied to us and the list is not in alphabetical order, your algorithm should output the evidence of this.

6. There exists a haunted maze which contains $n$ scare stations, with a designated starting station $s$ and a final station $t$. To model the haunted maze as a graph, there is a vertex for each scare station and a directed edge from one station to another if it is easy to walk between the two directly (note: because the owners of the haunted house place a restriction on which houses you can visit, the edge between the two scare stations is not bidirectional). Each scare station $v$ has a scare factor of $c(v) > 0$, where the higher $c(v)$ is, the scarier the scare stations are. Thus the graph has costs on the vertices rather than the edges. Shindler is a scaredy-cat, and wants to complete the maze by minimizing the total scare factor of the houses; in other words, he wants to find a path $P$ from $s$ to $t$ such that $\sum_{v \in P} c(v)$ is minimum.

Suppose you already have Dijkstra’s Algorithm implemented for directed graphs. Describe how you can use that implementation to solve this problem. For credit, your answer must involve creating a set of edge weights for the same set of vertices and edges such that a call to Dijkstra’s algorithm will produce the desired tree. Do not re-create a “Dijkstra-like” algorithm and do not modify the code of the existing Dijkstra’s Algorithm.
7. The Sorting Hat problem is as follows. I have a collection of $n$ hats and $n$ mannequins (statues that resemble a human and are used to model clothing and accessories). My goal is to put one hat on each mannequin head. Each hat is a distinct size, and no two heads are the same size. For each hat, there is exactly one head that the hat fits perfectly. I cannot compare two hats or two heads, nor can I inspect a single hat or a single head to determine its size.

I can, however, try to place a hat on a head. If I do so, I find out one of the following pieces of information:

- The hat fits the head perfectly.
- The hat is too small for the head.
- The hat is too big for the head.

We can then remove the hat from the head if we so choose.

The goal of the Sorting Hat problem is to match each head with the hat that fits it perfectly while placing as few hats on heads as possible. Give an approach to do so. If there are $n$ hats and $n$ heads, express the number of times your approach will place a hat on a head in $O$-notation. Clearly indicate if you are measuring worst-case or expected case (either is okay, but be sure to clearly indicate it).

8. Prove that the difference of any two odd perfect squares is always a multiple of 4.

9. Prove that if a number is 2 more than a multiple of 4, it is not a perfect square.

10. Recall the Fibonacci sequence, where $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. Prove that, for all non-negative integers $n$, $F_n$ is even if and only if $n$ is divisible by 3.

11. Suppose we seat 25 computer science majors and 25 philosophy majors (no double majors in this problem) around a circular table. Show that there is always a person seated with two philosophy majors adjacent to them.