1. In each part give a Big-Oh estimate of the worst-case time it takes to find a specific entry in the given structure
   a. An unsorted Linked List with \( n \) entries
      \( O(n) \)
   b. A sorted Array List with \( n \) entries
      \( O(\log(n)) \)
   c. A sorted Linked List with \( n \) entries
      \( O(n) \)
   d. A Binary Search tree with \( n \) entries
      \( O(n) \)
   e. An AVL Tree with \( n \) entries
      \( O(\log(n)) \)
   f. A Heap (Priority Queue) with \( n \) entries
      \( O(n) \)
   g. A directed graph with \( n \) Vertices and \( m \) Edges, using the graph structure from Lab 9 (so a Graph has a Hashmap of Vertices and a Vertex has a Linked List of Edges). Here the entries you are looking through are stored in the Vertices of the graph.
      \( O(n) \)
2. We might have a binary tree structure with integer nodes defined as
   class Tree {
       int data;
       Tree left, right;
   }

   Let’s assume missing or empty trees are represented as null and A leaf is a tree that has
   null left and right children. **Write a method int height()** for this Tree class that returns
   the height of the tree rooted at the current node. Leaves should have height 0, a tree
   with leaves for both children should have height 1, and so forth. Note that this is slightly
different from the code you wrote for Lab 5 because there is no EmptyTree class; you
have to handle the base case of the recursion yourself.

   int height( ) {
       if ((left == null) && (right == null))
           return 0
       else if (left == null)
           return 1+right.height();
       else if (right == null)
           return 1+left.height();
       else
           return 1+Math.max(left.height, right.height);
   }
3. Here is a picture of a binary tree with 10 nodes.

I am going to run an algorithm that uses a “structure” that might be a stack or a queue. I start by putting the root node A into the structure. Then at each step I remove the next node from the structure, print its letter, add its left child into the structure and then add its right child into the structure. This continues until the structure is empty.

A) In what order are the nodes printed if the structure is a stack? An answer might look like “AGHBCDFEIJ” (though that’s not the right answer).

   AFGJHIBDEC

B) In what order are the nodes printed is the structure is a queue?

   ABFCDGEHJI
4. I have a priority queue with 9 entries stored as a heap in the following array:

```
  0  1  2  3  4  5  6  7  8  9  10  11  ...
[5  30 10 40 35 15 20 50 60 13  ...]
```

The small digits on top are the array indices. I want to add entry 13 to this priority queue. What is the sequence of entries in the array after this addition?

Your answer should look like this: 5 30 10 40 35 15 20 50 60 13, though that isn’t the correct answer.

5 13 10 40 30 15 20 50 60 35
5. I want to build a Hashmap in an array of size 10 (so the array indices are 0 through 9). I will use linear open addressing to resolve collisions. Here is a table of entries and hash values:

<table>
<thead>
<tr>
<th>Entry</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash Value</td>
<td>0</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

I will add the data in the order: D B C I G A E F H

List the entries in the order they appear in the array: index 0 first, then index 1, and so forth.

G A E F H D I B C
Hackers have broken into Oberlin’s database and destroyed all grade information for last semester except for one backup file. Each line of the file has the format

```
student-name | class-name | grade
```

The lines of the file are in no particular order. You need to read this file and produce a list of every student and their gpa for the term, and also a list of all of the classes that were offered, the students in the class, and the grade for each student for the class. The first list might look like:

- Barney Rubble 2.1
- Fred Flintstone 3.2

and the second list might be

- CINE 101
  - Rocket J. Squirrel A
  - Bullwinkle Moose C-
  - Boris Badenov D
  - …
- CINE 102
  - Yogi Bear B
  - Booboo Bear B+
  - Mr. Ranger A-
  - …

How will you use data structures to solve this problem? **Your answer should be in English, not code.** In particular:

a. What will you do with each line of the file?

   We build a graph that has two kinds of nodes and two kinds of edges. There are student nodes and class nodes. Student nodes have weighted edges going to their classes; the grade in the class serves as the weight. Class nodes have unweighted edges to all of their students. The graph consists of two HashMaps: one mapping student names (strings) to a Student node structure, the other mapping class names to the Class node structure.

   When we read a line of the file in the form student | class | grade we separate the line into its three fields. If the student name is not one of our vertices we add a new vertex with this name; if the class name is not one of our vertices we add a class with this name. Then we add an edge from the class vertex to the student vertex, and a weighted edge from the student vertex to the class vertex with the grade as the weight.

b. How will you print the first list of students and their gpas?

   The Student HashMap will let us iterate through the keys, which are the student
names. For each name we print the name and then go to the vertex for that name and walk through its edges, summing the grade values and counting how many there are. After going through all of the edges we print the average.

c. How will you print the second list of classes, their students and grades? The Class Hashmap will let us iterate through the class names. For each class there is an edge going to each of its students so we can print the student’s name. From the student vertex we have to look through all of the edges to classes to find the edge back to the class so we know the student’s grade in the class. (If you don’t like this, make the edge from class to student also have a copy of the grade. That takes up more space but would give faster runtime.)
7. We might have a Node class defined as

```java
class Node {
    private int data;
    private Node next;
}
```

and a List class with an empty sentinel node Head at the front and a variable Tail pointing to the last node in the list. Here is a picture of such a list containing the data 10, 20, 30:

```
Head → 10 → 20 → 30 → Tail
```

Give a method `List reverse(List L)` that will build the reversal of List L. So if L starts as the list shown, `reverse(L)` will produce the list 30, 20, 10. It is acceptable to destroy L in the process of making its reversal (for example, you might want to change some of the links in L).

```
List reverse(List L) {
    if (L.size() < 2)
        return L;
    else {
        Node p = L.Head;
        Node q = p.next;
        Node r = q.next;
        Node newTail = q;
        while (p != L.Tail) {
            if (p == L.Head)
                q.next = null;
            else
                q.next = p;
            p = q;
            q = r;
            if (r != null)
                r = r.next;
        }
        L.Head.next = L.Tail;
        L.Tail = newTail;
        return L;
    }
}
```