CS 211 **Computers and Programming** Handout xx
Fall 1997

Assignment 8: Data Structures

November 19th, 1999

Due date: December 2nd, 1999

1. (More big-O complexity) Scarlett O'Java was not beautiful but men seldom realized it when caught by her mastery of big-O notation as the Tarleton twins were. They were specially intrigued by Scarlett's claim that in big-O complexity, it is immaterial what the base of a logarithm is. In particular, she claims that if an algorithm is $O(log_3(n))$, it is also $O(log_2)(n)$ and vice versa. Argue that she is correct by showing the following:

   - $log_2(n) = O(log_3(n))$
   - $log_3(n) = O(log_3(n))$

   In each case, you must give us a witness (that is, a (c, N_0) pair) as required by the formal definition of big-O notation.

2. (Heaps) The eponymously named Uriah Heap is trying to understand heapsort. Assuming that the heap is maintained as a *complete binary tree* as discussed in class, draw a sequence of pictures that shows the heap during the heapsort process if the input values are $4, 3, 8, 7, 2, 9$ in that order (so the first element inserted into the heap is 4 etc.). What is the total number of comparisons made by heapsort to perform the sort? What is the total number of comparisons performed by selection sort for this example? You may want to check your answer (and my code) by executing the heapsort program we put online.

3. (Heaps) Little Bo Peep has lost her heap which was a complete binary tree of height h. What are the smallest and largest number of items she could have kept in this heap?

   Suppose she has n items to store in the heap. What is the height of the heap?

4. (HeapAsArray) In class, we showed you that you can use an array to store a heap. Reimplement the Heap class given online in the lecture notes so that it uses an array of Comparables rather than a tree. Turn in your code and show us the output produced when you run the Uriah Heap heapsort example with this heap.

   Note: Your class should implement a constructor that takes the maximum size of the heap as an integer parameter and allocates an array of the appropriate size. You should report an error if the heap overflows when too many elements are inserted into it.

5. (BST) Assume that we are given an empty Binary Search Tree (BST). With the help of a sequence of drawings, show the result of inserting the values $3, 1, 4, 6, 9, 2, 5, 7$ in that order. Then show the BST that results when the root is deleted.

6. (BST) Given a BST, we are interested in printing all elements that are in the range specified by the two keys, low and high.

   - Write a method that accomplishes this task by using an auxiliary recursive method. Turn in your code.
   - What is the asymptotic complexity of the running time of your algorithm?

7. (Hashtables and Heaps) Dean Malthus would like to alleviate overcrowding in grad student offices, so he wants to figure out which offices are the most crowded, and who the students are in those offices. He has a file that contains student names and office numbers (the format of this file is explained later). He wants you to write a program that will let him compute the m most crowded offices, where m is a positive integer that he will pass as a parameter to your program.

   Write a program for doing this, following these guidelines.

   1. The value of m, and the name of the file containing student offices must be command line parameters to your program.
   2. The input file will contain names and offices in the following format.

      ```
      Eric Aaron       4157
      Marcos Aguilera  5151
      ...
      ```

      You may assume that all students have a first name followed by a last name, and that the office number is a positive integer. Each student entry will be on a different line.
   3. Your program must print the m most crowded offices, together with the names of the students in these offices. The most crowded office must be printed first.
      There may be multiple offices with the same number of students. If you print one of these offices, you must print *all* offices with the same number of students, so you may end up printing more than m offices. For example, suppose $m = 2$, and the occupancy of offices is the following:

      ```
      4157 4
      5151 4
      5150 4
      213  3
      221  3
      110  2
      ```

Your program must print the office numbers 4157, 5151, 5150 together with their occupants.

4. The first order of business is simply to figure out how many students there are in each office. We suggest you read in each student name and office number from the file, and insert this information into a hash table, using the office number as a key. If there is already an entry in the hash table with this key, you simply add the student name to the other names for this office in the hash table. You can use one of the hash table implementations in Java.
5. To find the m most crowded offices, we can use a heap. Enumerate the entries in the hash table, and enter each entry into the heap, using the number of students as the priority. To find the m most crowded offices, you can do m `get` operations on the heap. You can use the heap implementation we put online.
6. Test your program for the data files given to you online for $m = 4$, and $m = 3$, and turn in the output produced by your program.

What is the asymptotic complexity of your algorithm?