**due Sunday, 24 February 2019, at 11:00 PM**

*Note: If you choose to work with a partner on this homework, you need to register by 5:00 PM on January 18. Instructions for doing so will be given later, but you will need to follow them exactly. You are allowed to work with a partner in a different section.*

*When you submit your assignment solutions please take the time to indicate the starting page of each answer. To make life easy for the TA's put the answer to each problem on a separate page (you may put multiple parts of the same problem on one page)*

*If you are working with a partner, please submit **one solution only**. GradeScope allows you to specify your partner after a submission. Moodle does not, but we will grade **one solution per team only.***

Submissions for the written part of the homework, including your analysis of the programs should be submitted to GradeScope (these will be set up as two distinct assignments – `Homework 2` and `Program 2 Analysis`). The source files for your programs should be submitted to `Assignment 2` in Moodle.

1. [9 points: 3 for (a), 4 for (b), and 2 for (c)] *Purpose: Understanding Heapsort and lower bounds on algorithms.*

   Consider the special case where Heapsort is used to sort 0's and 1's. Because Heapsort is comparison-based, you should assume that Heapsort does not take any special advantage of the values of the keys. Let $k$ be the number of 1's in the array. All bounds that follow are expressed as functions of *both* $n$, the total number of elements, and $k$, the number of 1's. Assume $k < n/2$. Recall that the MakeHeap phase takes linear time, so no need to say anything about it in your proof.

   (a) Prove that the worst case number of key comparisons in the situation described above is $O(n + k \lg n)$. This is linear unless $k \in \omega(n/\lg n)$.

   (b) Show that the above bound is 'tight' in the sense that it is also a $\Omega$ bound for the algorithm. Recall that to prove a worst case lower bound of $\Omega(g(n))$ for an algorithm we need to show that there exists $c, n_0 > 0$, so that for each $n \geq n_0$, there exists an input of size $n$ that causes the algorithm to take time (or, in this case, number of comparisons), $\geq cg(n)$. Where a time bound has two parameters, the argument needs to work for any valid combination of the two.

   (c) Prove that the number of key comparisons depends on the original position of the 1's in the array? This is not about the worst case or an asymptotic bound. The question refers to how the exact number of comparisons depends on the positions of the 1's. Since your proof will require an example with duplicate keys, you should use key,value pairs[1] to distinguish between elements. Make your example as small as possible.

[1] See the programming assignment below.

2. [10 points: 2 for (a), 3 for (b), and 5 for (c)] *Purpose: understanding sorting algorithms and the sorting lower bound.* Problem 8-4 on pages 206–207 (179–180 in 2/e): red and blue jugs.

3. [6 points, 3 points each part] *Understanding the analysis of linear time selection.* Exercise **9.3-1** on page 223 (page 192 in 2/e): selection with groups of size other than five.