Note: The homework is due electronically on Gradescope and Canvas on Sunday, October 21 by 6:00 pm EDT. You may NOT use any late days on this homework; solutions will be distributed at the midterm review session on Sunday to help you study for Midterm 2.

A. Gradescope: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Failing to do so will get you 5% off, which cannot be argued against after the fact. Gradescope may prompt you with a warning to select your cover page, please ignore this warning.

B. Canvas: You must also submit your assignment on Canvas. Forgetting to do so will incur a 10% penalty.

C. \LaTeX{}: You must use the `hw121.cls` Latex template provided on the course website, or a 5% penalty will be incurred. Handwritten solutions or solutions not typeset in Latex will not be accepted.

D. Solutions: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements. Piazza will also contain a complete sample solution.

E. Algorithms: Whenever you present an algorithm, your answer must include 3 separate sections. Please see Piazza for an example complete solution.

1. A precise description of your algorithm in English. **No pseudocode, no code.**
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

F. Collaboration: You are allowed to discuss ideas for solving homework problems in groups of up to 3 people but you must write your solutions independently. Also, you must write on your homework the names of the people with whom you discussed. For more on the collaboration policy, please see the course webpage.

G. Outside Resources: Finally, you are not allowed to use any material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you’re unsure if something violates our policy, please ask.
Midterm 2 Topics:

- Topics covered in Midterm 2 build off concepts covered in Midterm 1
- Trees: Binary, Binary Search, Self-Adjusting (Splay), Balanced BSTs (Red-Black)
- Priority Queues and Heaps
- Homework Assignments 3–5
- There may be a programming question on the midterm exam

This homework assignment has questions from previous CIS 121 exams, as well as questions similar in nature to those that will be on Midterm 2. This assignment is meant to help you prepare for the upcoming midterm — during the review session on Sunday, October 21 from 6-8pm, we will be going over the solutions to some of these problems, as well as going over topics. Solutions will also be distributed at the review session.

1. [8 pts] Answer the following in 1-2 sentences. Solutions longer than 2 sentences will not be given full credit.

   a. A binary tree is a tree where every node has at most two children. We can generalize this to $c$-ary trees where each node has at most $c$ children where $c \geq 2$. Give an upper and lower bound on the height of a $c$-ary tree with $n$ elements (the bound should be exact, i.e., don’t use $O$ or $\Theta$ notation).

   b. What impact does the height of a tree have on the worst case running time for standard search tree operations?

   c. When does the black height of a red-black tree change?

   d. Which of the data structures or algorithms we learned so far in class would be most practical to use in each program. Give a 1 sentence description why you picked that one.
      
      i. Given a stream of a trillion numbers (which is too big of a number to store in memory at one time), keep track of the $k$ highest numbers in a space efficient way, where you can assume $k$ numbers can easily fit into memory.

      ii. You have a database of billions of keys, but only 10% of which are accessed frequently, and you want to enable fast look-ups.
2. [7 pts]

a. Suppose that we have the numbers \{1, 2, ..., 1000\} in a binary search tree and want to
search for the number 363. Which of the following sequences could not be the sequence
of nodes examined? Give a 1 sentence explanation for why not.

   (i) 2, 252, 401, 398, 344, 397, 363
   (ii) 924, 220, 911, 244, 898, 258, 362, 363
   (iii) 925, 202, 911, 240, 912, 245, 363
   (iv) 2, 399, 387, 219, 266, 382, 381, 278, 363
   (v) 935, 278, 347, 621, 399, 392, 358, 363

b. Suppose the search for key \(k\) in a BST ends up in a leaf. Define the following sets:

   \(L\) — the keys to the left of the search path
   \(S\) — the keys on the search path
   \(R\) — the keys to the right of the search path

   Prove or provide a counter-example: any three keys \(l \in L, s \in S, r \in R\) must satisfy
   \(l \leq s \leq r\).

3. [8 pts]

a. Assign the keys 2, 3, 5, 7, 11, 13, 17, 19 to the nodes of the binary search tree below so
   that they satisfy the BST property.

b. Prove that this tree cannot be colored to form a legal red-black tree (hint: which red-
   black BST property cannot hold for this tree?)

c. It turns out that there is a single rotation you can perform on this tree such that it can
   be colored as a valid red-black tree. Perform this rotation and draw the red-black tree
   that results (with colors).
4. [8 pts] You are given the following Node class:

```java
class Node {
    int key;
    Node left, right;
    /* constructor omitted */
}
```

Implement the following two methods, which will be used in your Splay Tree class. For example, `rotateRight(node Q)` should return node P. Likewise, `rotateLeft(node P)` should return node Q.

```java
Node rotateRight(Node n) {
    // TODO: implement
}
```

```java
Node rotateLeft(Node n) {
    // TODO: implement
}
```

You are expected to write code for this question, this is an exception to our “no code on homework assignment” rule. If you think it would help to explain your logic, feel free to include a written explanation in addition to your code. Please format your code on LaTeX using \begin{verbatim} ... \end{verbatim}.

5. [9 pts] Suppose you are given a BST and a min-heap and you are given an entry that exists in both data structures. You want to find the rank of the entry. An item is of rank k if it is the k-th smallest value in its data set.

a. In what runtime could you find the rank of the entry in the BST? Why? (Note: you do not need to provide a full algorithm).

b. Assume that you changed the BST so that each node contained the total count of all of its descendants. How would that change the runtime? How would your answer change if the tree was balanced?

c. Can you find the rank of this element in the same runtime in your min-heap? Why or why not?