Project 2 provides experience with binary tree operations and experimental analysis of algorithms. You will design, implement, and verify a program that experimentally measures time to perform insertion and deletions with binary search trees. The experiment has the following 6 experimental conditions (operations) and 12 dependent variables (measurements).

<table>
<thead>
<tr>
<th>tree</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>create (2^{10}-1 nodes) delete (2^{9}-1 nodes) insert (2^{8}-1 nodes)</td>
</tr>
<tr>
<td>balanced</td>
<td>average and max level average and max level average and max level</td>
</tr>
</tbody>
</table>

You will design and develop classes for a binary search tree of Elements and for your experiment's statistics. For simplicity Elements is a class containing Integer "key" values. Your binary search tree implementation must use reference variables to reference the left and right subtrees. You can not implement the binary search tree in an array or ArrayList. In the experiment Elements will be added to a tree in two orders: (1) random, or (2) balanced. A balanced order could be generated by inserting Elements from an ArrayList<Element> that is created (sorted) in ascending order by modifying a recursive binary search algorithm. Random order insertion could be from the initially ordered ArrayList that has been randomly shuffled (each value has been placed in a randomly selected index of the ArrayList). Note: you have a linked binary search tree and you have an ArrayList. The ArrayList is used to hold the Elements you are going to insert and delete into the binary search tree. The ArrayList is only used to hold the values that will be inserted at creation of the binary search tree, a subset that will be deleted from the tree, and last a smaller subset that will be re-inserted into the tree.

In the experiment there will be 20 samples (trials, runs) of each experimental condition. Given the 6 experimental conditions (above table) there will be 40 trees created (20 random and 20 balanced). First a tree is created. Second, nodes are deleted from the tree. Third, nodes are re-inserted into the tree. After each experimental operation, the average level and max level of the resulting tree will be measured. Thus, twelve tree levels (6 average and 6 maximum) will be measured. The tree levels are the "dependent variables" in the experiment. The six experimental conditions are the "independent variables". In the delete experimental condition, the first 511 (2^{9}-1) Elements from the randomly ordered ArrayList are deleted (actual, not dummy deletion) from the tree. In the subsequent insertion condition 255(2^{8}-1) Elements are re-inserted back into the tree. Each experimental condition is run, or sampled, 20 times. That will generate 20 average levels and 20 maximum levels for each experimental condition. The average of these samples will be computed and scaled by the number of operations. For example, the averageLevel/node for a "balanced tree" sample's creation = createAverageLevel/ 1023. This would be that experimental condition's average level result.

After you have designed and implemented a correct generic binary search tree data structure you can conduct your experiment -- run the program to generate your experimental results. When you have your results you should write a brief report (with table of data) presenting and discussing your
dependent variables. You must compare your experimental results with expected results where appropriate.

If you are uncertain about using generics, you could first design and implement with Integers (not ints). This would give you the results you need for your submission. If you have time you could then convert your code to generics. You can submit a non-generic assignment, but you will not earn the credit that an equivalent generic assignment would.

You could plan the design and implementation of this project in four phases. In phase 1 you would design and implement a correct binary search tree data structure for Integers. You could test this with a small number of nodes (say 7). You could create, delete one from each level, and insert the deleted values back. Once you have a correct binary search tree you could create the first draft of your UML diagrams. Next, in phase 2 you could design and implement the experiment. You could first test with your small sample (7) to see that your results are valid and then use the assignment number of nodes. This would generate appropriate data for you. You could then do the third phase; write the report about your experimental results. Last you could modify the software to work with generics and update your UML diagrams (phase 4). This "plan" is presented to support the idea of using a "solve a smaller problem first" approach. You could have a different design / implement / test plan.

Submit your source code, UML "lite" class diagrams for your classes, a brief description overview of, or, orientation to, your software's design, and a brief report discussing the performance aspects of your experiment. You should "zip" or "jar" your files into a single file that you can submit on Moodle.

Manage your time well, this project requires more design and analysis on your part. As with project 1 the lab is a good time to work on the design and implementation of the project. This is the time for one-on-one question/answering with me, or your group. Remember to learn what everyone in the group knows, this material is also on the exams.

"Plan well and prosper."