Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to analyze/evaluate the time complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the exercises described below.

Exercise 1 (35 points)

Suppose that we have numbers between 1 and 1200 in a binary search tree (BST), and we want to search for the number 400. Which of the following sequences could **not** be the sequence of nodes examined on a BST? (*Recall Grading Guidelines. I suggest to use tree drawing to justify your answer. Drawing is strongly advised to make explanation easier to follow for you and your grader. Justify only for the sequence(s)that could* ***not*** *be the sequence of nodes examined.*)

a. 961, 257, 948, 281, 935, 295, 399, 400.

b. 1, 398, 388, 220, 267, 383, 382, 304, 400.

c. 961, 304, 373, 617, 325, 418, 384, 400.

d. 951, 228, 937, 266, 938, 282, 400.

e. 39, 289, 438, 435, 367, 381, 434, 400.

First I will say I am using the pseudo code from the book for the TREE-SEARCH algorithm:

TREE-SEARCH(x, k)

1 if x == NIL or k == x.key

2 return x

3 if k < x.key

4 return TREE-SEARCH(x.left, k)

5 else return TREE-SEARCH(x.right, k)

1. 961, 257, 948, 281, 935, 295, 399, 400

This sequence could NOT be a valid using TREE-SEARCH in a Binary Search Tree as follows.

Starting with the beginning of the sequence 961 and searching for 400. Since 400 is less than 961 we will return x.left to 257. Now with 257 being the new x we compare again to are search number 400. Since 400 is greater than 257 we will return x.right. So going right to 948. Then we compare 948 to 400, which is it less than. So we will go left again to 281. We then compare 281 to 400, which it is greater than, so we go to the right child of 281. This node is 935. Comparing 935 to 400 we have it being less than so then we move to the left child of 935 which is 295. Then again comparing 295 to 400 we see that 400 is more, so we move to the right child node which is 399. Then we compare 399 to 400 which is less than so we should be moving to the left child node of 399 which is 400. This is not valid as the left child node of 399 should be less than 400.

1. 1, 398, 388, 220, 267, 383, 382, 304, 400

Starting with the beginning of the sequence we compare the number to be searched which again is still 400. Starting at 1 in the sequence we compare 400 with 1. Since 400 > 1, we return x.right to 398. So, 398 is the right child of 1. Continuing at 398 we compare 400 with 398. Since 400 > 398, we return x.right to 388. So then this should mean that 388, which is next in the sequence, would be the right child to node 398, which cannot be true in a Binary Search Tree as 388 is less than 398 and would make it the left child.

c. 961, 304, 373, 617, 325, 418, 384, 400.

This is considered a valid BST sequence and does not break any BST property, and no other explanation or justification is needed as the description said.

1. 951, 228, 937, 266, 938, 282, 400.

This would not be considered a valid binary search tree sequence as follows. Starting with the beginning of the sequence, we compare the number to be searched, which again is 400. Starting with the first node in the given sequence 951. We compare 400 with 951. Since 400 < 951, we return x.left to 228. So, 228 is the left child of 951. Then at node 228 we repeat the same cycle. We compare 400 with 228. Since 400 > 228, we return x.right to 937. So, 937 is the right child of 228. We compare 400 with 937. Since 400 < 937, we return x.left to 266. So, 266 is the left child of 937. We compare 400 with 266. Since 400 > 266, we return x.right to 938. So, 938 is the right child of 266. This is where the sequence becomes invalid. In a BST, the right child of 266 should be greater than 266 but also should be less than 937. However, 938 is greater than 937, which violates the BST properties. Therefore, the sequence 951, 228, 937, 266, 938, 282, 400 is not possible.

e. 39, 289, 438, 435, 367, 381, 434, 400.

The sequence does not break any BST properties at any point and adheres to the TREE-SEARCH algorithm. Therefore, the sequence 39, 289, 438, 435, 367, 381, 434, 400 is valid and no further justification is needed as the description said.

Exercise 2 (65 points) Build a Binary Search Tree

Starting with an empty binary search tree , insert in the following numbers (strictly in this order): 900 1050 165 160 137 500 600 770 832 300

Whenever you insert one element (using TREE-INSERT from the lectures/textbook), show the resulting new binary search tree (draw the tree). It is ok to draw by hand the resulting new binary search tree, take pictures, insert.... Just make sure your handwritten/drawn trees are neat (neatness is worth 15%). We should see 10 separate "*growing*" binary search trees.

Starting with an empty Binary Search Tree we insert 900 as the root of the tree as it is the first value in the sequence given. The graph looks as follows after:

A grey circle with black text

Description automatically generated

**To insert 1050, we first compare it with 900. Since 1050 > 900, insert 1050 as the right child of 900. The resulting binary search tree is as follows after:**

**A graph of binary search results

Description automatically generated**

**Then since the left child of 900 is still null we compare 900 again with the next node in the sequence which is 165. Compare 165 with 900. Since 165 < 900, insert 165 as the left child of 900. The result graph is as follows:**

**A graph of binary search results

Description automatically generated**

**Then to insert 160 we compare 160 with 900. Since 160 < 900, go to the left subtree. Again compare 160 with 165. Since the left and right child of 165 is null, meaning available for insertion. Since 160 < 165, insert 160 as the left child of 165. The resulting graph is as follows:**

**A graph of binary search tree

Description automatically generated**

**To insert 137, we first compare it with 900. Since 137 < 900, go to the left subtree. Compare 137 with 165. Since 137 < 165, go to the left subtree again. Compare 137 with 160. Since the left and right child of 160 are null, meaning available for insertion. Since 137 < 160, insert 137 as the left child of 160. The resulting graph is as follows:**

**A graph with numbers and circles

Description automatically generated**

**To insert 500, we first compare it with 900. Since 500 < 900, go to the left subtree. Compare 500 with 165. Since 500 > 165, go to the right subtree. Since the right child of 165 is null, insert 500 as the right child of 165. The resulting graph is as follows:**

**A diagram of numbers and circles

Description automatically generated**

**To insert 600, we first compare it with 900. Since 600 < 900, go to the left subtree. Compare 600 with 165. Since 600 > 165, go to the right subtree. Compare 600 with 500. Since 600 > 500, insert 600 as the right child of 500. The resulting graph is as follows:**

**A diagram of a tree

Description automatically generated**

**To insert 770, we first compare it with 900. Since 770 < 900, go to the left subtree. Compare 770 with 165. Since 770 > 165, go to the right subtree. Compare 770 with 500. Since 770 > 500, go to the right subtree. Compare 770 with 600. Since 770 > 600, insert 770 as the right child of 600. The resulting graph is as follows:**

**A graph with numbers and lines

Description automatically generated**

**To insert 832, we first compare it with 900. Since 832 < 900, go to the left subtree. Compare 832 with 165. Since 832 > 165, go to the right subtree. Compare 832 with 500. Since 832 > 500, go to the right subtree. Compare 832 with 600. Since 832 > 600, go to the right subtree. Compare 832 with 770. Since 832 > 770, insert 832 as the right child of 770. The resulting graph is as follows:**

**A graph with numbers and circles

Description automatically generated**

**To insert 300, we first compare it with 900. Since 300 < 900, go to the left subtree. Compare 300 with 165. Since 300 > 165, go to the right subtree. Compare 300 with 500. Since 300 < 500, insert 300 as the left child of 500. The resulting graph is as follows:**

**A graph with numbers and circles

Description automatically generated**

What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

* **Student 1** answers: "9:48am". Student 1 will get 25 points.
* **Student 2**answers : "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
* **Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (**document is attached**).



Therefore, the time t = 108 miles/60mph \* 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".