# CST 370 – Spring A 2020

**Homework 5**

## Name: Randy Son

Class ID: 7774

**How to turn in?**

* Write your answer to the questions 1 to 8, and submit it on the iLearn. You can submit the file in PDF format. Don’t forget to write your name and class ID number at the beginning of the file.
* For Question 9, you should submit your C++ source file on the iLearn.
* Thus, you have to submit two files (one PDF file and one C++ source file) on the iLearn.
* Note that the due date is 11:55(PM). This is the iLearn’s timestamp, not your submission time. Since there could be a long delay between your computer and iLearn, you should **submit early**.

[**Note**: If it is difficult to draw diagram(s) when doing homework, you can draw them on paper by hand. Then take a picture of the paper and insert it.]

1. (5 points) The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Which one of the following is the postorder traversal sequence of the same tree? Explain your answer.

a) 10, 20, 15, 23, 25, 35, 42, 39, 30

b) 15, 10, 25, 23, 20, 42, 35, 39, 30

c) 15, 20, 10, 23, 25, 42, 35, 39, 30

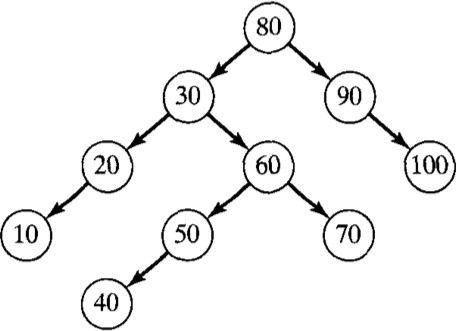
d) 15, 10, 23, 25, 20, 35, 42, 39, 30

The root in the preorder traversal sequence is 30, which should be last in the postorder traversal sequence. Given that the root is 30, values less than thirty would be on the left and values greater than thirty would be on the right.

20, 10, 15, 25, 23 | **30** | 39, 35, 42

The least value on the left is 10, which indicates that it can be the last, or that it has a child which is the furthest from the root. 15 makes sense, which implies 15, 10 is a viable path to begin at. From this, 23 must be a child of 25.

1. (10 points) Consider the following Binary Search Tree (BST):



a. Perform an inorder traversal of this BST.

10 -> 20 -> 30 -> 40 -> 50 -> 60 -> 70 -> 80 -> 90 -> 100

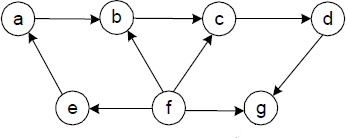
b. Perform a preorder traversal of this BST.

80 -> 30 -> 20 -> 10 -> 60 -> 50 -> 40 -> 70 -> 90 -> 100

c. Perform a postorder traversal of this BST.

10 -> 20 -> 40 -> 50 -> 70 -> 60 -> 30 -> 100 -> 90 -> 80

1. (5 points) Apply the source-removal algorithm to solve the topological sorting problem for the following digraph. You should explain your answer and present the topological order clearly.



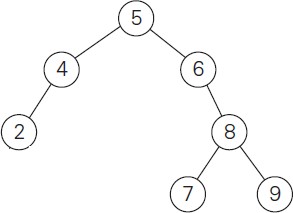
First, we remove **f** as it is the source in this graph with no incoming edges. Once **f** is gone, **e** is the next best choice as it has no incoming edge, but has an outgoing edge to continue the path.

f -> e -> a -> b -> c -> d-> g

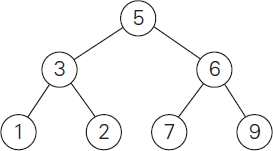
1. (10 points) A detachment of *n* soldiers must cross a wide and deep river with no bridge in sight. They notice two 12-year-old boys playing in a rowboat by the shore. The boat is so tiny that it can only hold two boys or one soldier. Note that the boat needs at least one boy or one soldier to pass from one shore to another.
2. How can the soldiers get across the river?
3. First the two boys make it across the river
4. 1 boy takes the boat back and 1 boy stays across the river
5. 1 soldier takes the boat across
6. The boy across the river takes the boat back
7. Repeat #1 until all soldiers have crossed
8. How many times need the boat pass from shore to shore?

4**n** times, where **n** is the number of soldiers. It takes 4 passes for 1 soldier to cross.

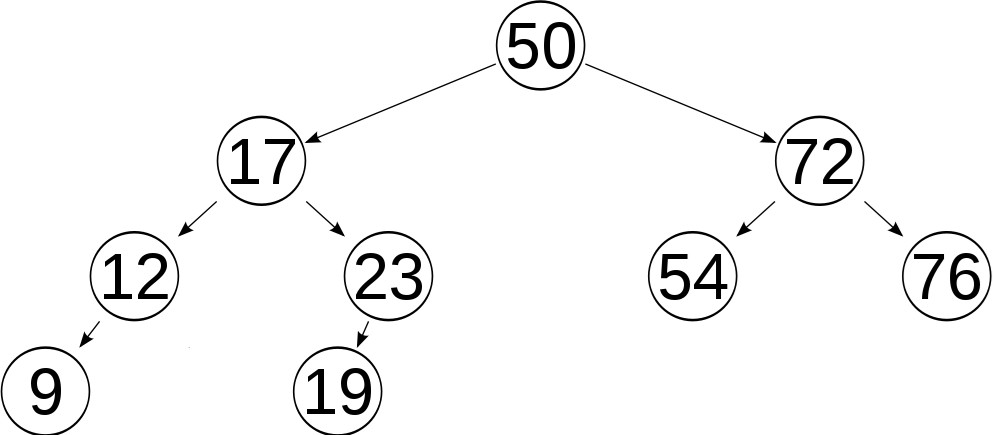
1. (10 points)
2. Is this an AVL tree? (Yes/No)



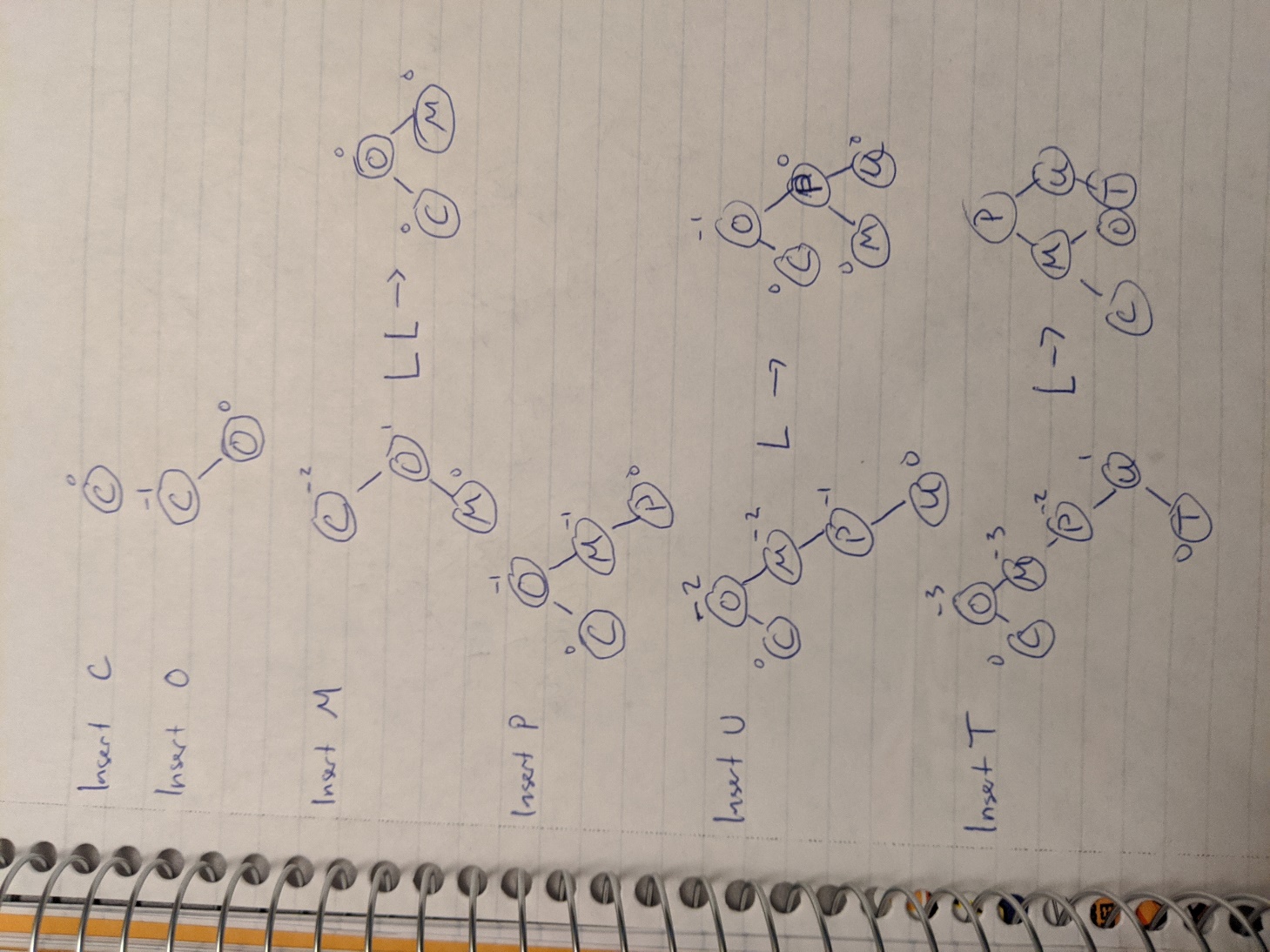
1. Is this an AVL tree? (Yes/No)

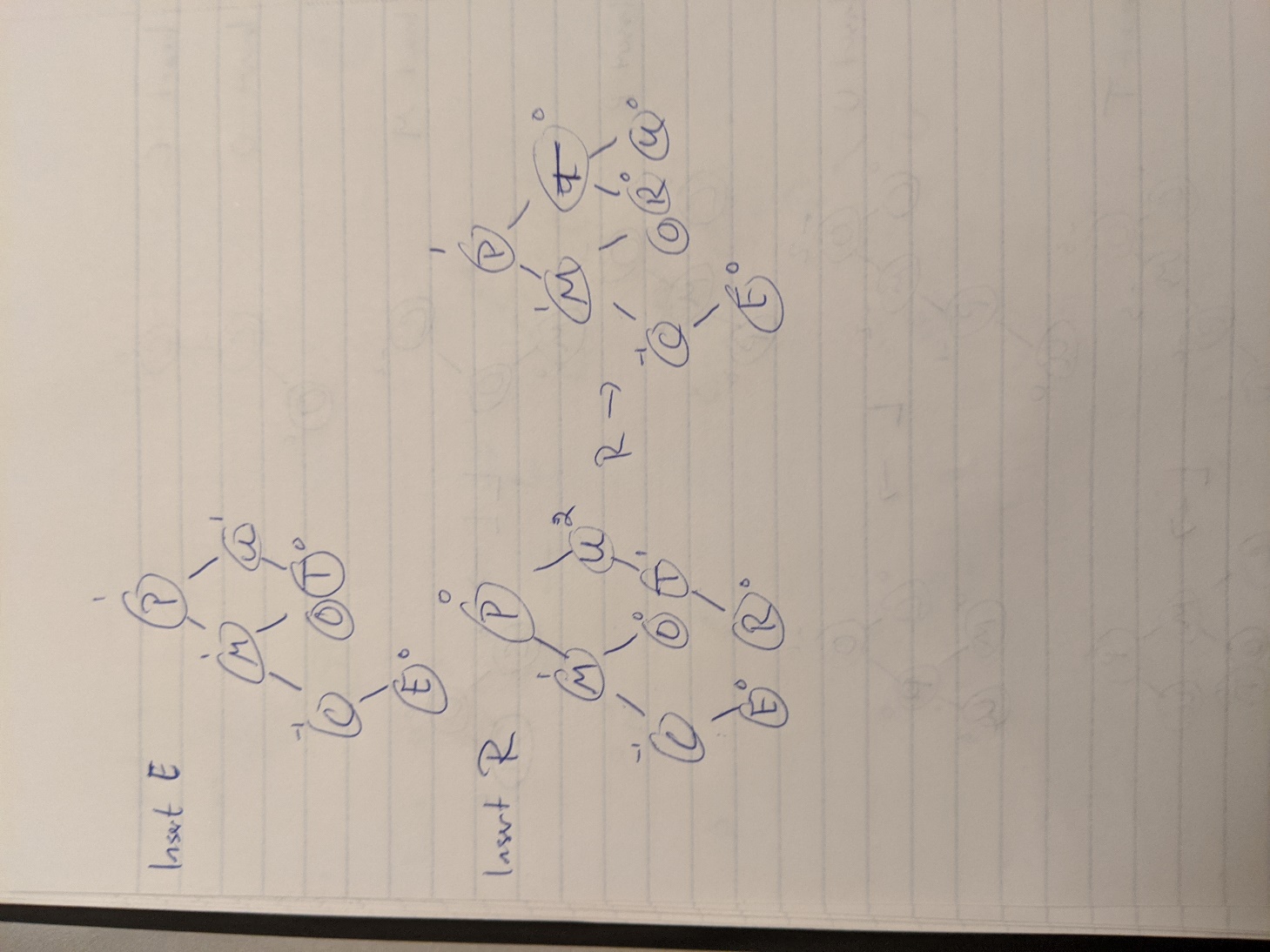


1. Is this an AVL tree? (Yes/No)



1. (10 points) Construct an AVL tree for the list **C, O, M, P, U, T, E, R**. Use the alphabetical ascending order of the letters and insert them successively starting with the empty tree. Your answer should present the rotation operations clearly for each letter addition.





1. (5 points) Suppose you are given a list of *N* integers. All but one of the integers are sorted in numerical order. Identify a sorting algorithm from class which will sort this special case in *O*(*N* ) time and explain why this sorting algorithm achieves *O*(*N*) runtime in this case.

Insertion sort would be the most efficient in terms of sorting with mostly sorted data. The reason for this is because we can search for the element that is out of place which at most takes O(N) time. Once we have located it, we only have to perform the insertion sort once, comparing our one element with at most n-1 other elements. Because of this, the time complexity for this special case is O(N).

1. (15 points) Let *A[0..n-1]* be an array of *n* integer numbers. Assume that all the numbers are distinct. In the array, a pair of two numbers (*A[i]*, *A[j]*) is called an ***inversion*** if *i < j* and *A[i] > A[j]*.
2. Assume that the array size is 4. What is the largest number of inversions possible in the array? Present a sample array with 4 integer values and describe your answer clearly.

The largest number of possible inversions is 6.

Sample array A: {4, 3, 2, 1}

pos *0, 1, 2, 3*

Inversions would be {{4, 3}, {4, 2}, {4, 1},

{3, 2}, {3, 1},

{2, 1}}

1. Similarly, answer the same question for an array with 6 integer values.

The largest number of inversions is 15.

Sample array A: {6, 5, 4, 3, 2, 1}

pos *0, 1, 2, 3, 4, 5*

Inversions would be {{6,5}, {6, 4}, {6, 3}, {6, 2}, {6, 1},

{5, 4}, {5, 3}, {5, 2}, {5, 1},

{4, 3}, {4, 2}, {4, 1},

{3, 2}, {3, 1},

{2, 1}}

1. Based on your answers to the question (a) and (b), what is the largest number of inversions in the general array with *n* elements?

For array size n, total number of inversions possible is (n\*(n-1)) / 2.

1. (30 points) Write a C++ program called **ts.cpp** that implements the ***topological sorting algorithm based on the source-removal algorithm***. Your program should read an input file name. Then it should display the topological ordering. In the problem, you can assume that the number of vertices in the input file is less than 50. Additionally, you can assume that the input graph is always a DAG (= Directed Acyclic Graph). When you write the program, don’t forget to include “Title”, “Abstract”, “ID (A four- digit number)”, “Name”, and “Date”.

**Input file format**: This is a sample input file called **t1.txt**.

3

0 1 0

0 0 1

0 0 0

The first line (= 3 in the example) indicates that there are three vertices in the graph. For the homework, we can assume that the first vertex starts from the number 0. Thus, **t1.txt** describes a directed graph like below:



0

1

2

One blank space is used to delimiter the data. Note that there’s no blank space at the end of each line. **If your program does not read the file properly, your program will get no credit.**

The following presents a sample run of the program. Your program should be compiled and executed exactly like this.

Enter filename: **C:\\tmp\\t1.txt**

Topological sort: 0 -> 1 -> 2

In the program, your program has to **follow our convention (= ascending order)**. This is another sample input file called **t2.txt**.

5

0 0 1 0 0

0 0 1 0 0

0 0 0 1 1

0 0 0 0 1

0 0 0 0 0

This is a sample run:

Enter filename: **C:\\tmp\\t2.txt**

Topological sort: 0 -> 1 -> 2 -> 3 -> 4

Again, your program must **follow our convention (= ascending order)**. So, your program should remove the vertex 0 first between the two source vertices 0 and 1.