PROGRAMMING ASSIGNMENT 4

Subject : Trees
Advisor : Asst. Prof. Dr. Burcu CAN, Asst. Prof. Dr. Adnan ÖZSOY, Assoc. Prof. Dr. Sevil ŞEN, Res. Assist. Burçak ASAL
Programming Language : C
Due Date : 24.12.2017

Introduction: By the help of this experiment, students will learn the basics of tree concept and it's fundamental operations such as search, insertion, deletion, listing.

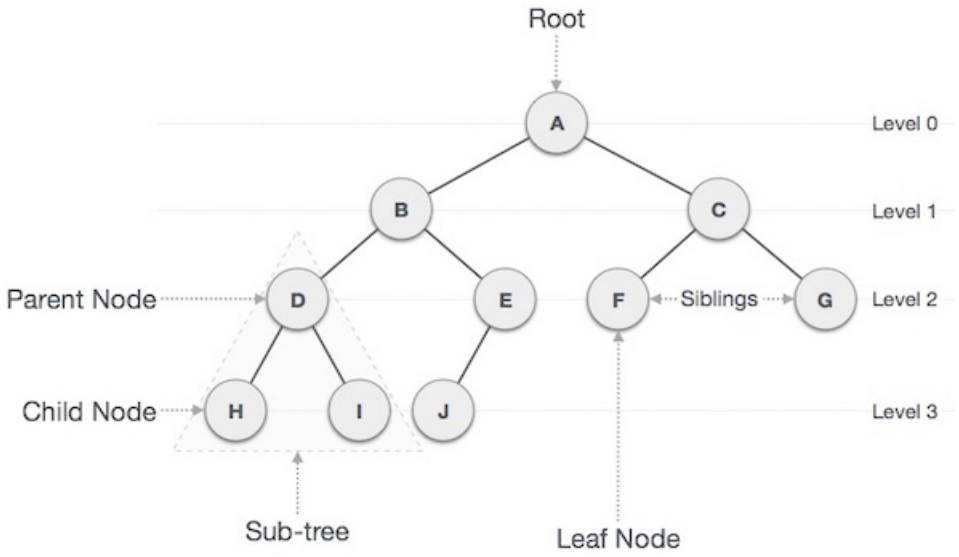

Figure 1: Schema of a basic tree concept

In this assignment, firstly, you will construct a tree (similar to the tree concept in Figure 1) with respect to the given first input file. Then, you will perform search, deletion and listing operations on the tree you constructed.

Part 1: Constructing the Tree

In this part, you will construct a tree for the given first input file. The first input file includes two columns (See Figure 2, left most). The first column consists of a group of unique numbers to be added as nodes in the tree. Each line in the second column specifies that how many lines you will read in the first column and add them as leaf nodes to the previous child nodes you added before.

During the construction of the tree, you should follow the constraints below (See also Figure 3):

1. During the reading process of the lines in the first column, with respect to the second column, if all the unique numbers in the first input file are inserted, then you must stop the reading of the first input file and ignore the rest of the second column (Figure

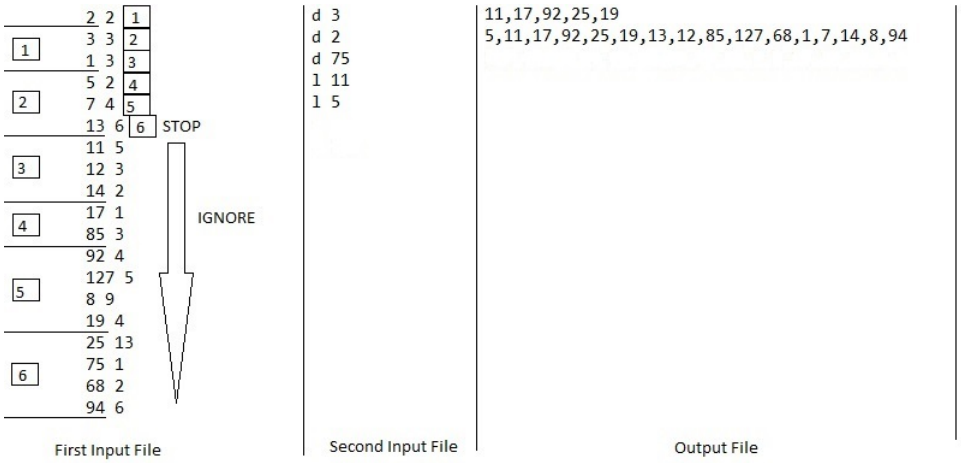

Figure 2: Input-Output File Formats for the Program

2, left most). Besides, if the last stopping value in the second column is bigger than the count of remaining lines/numbers then you must just read the remaining unique numbers without further processing.

2. During the insertion process, you must insert the unique numbers as new leaf nodes one by one to previous child nodes of the tree with the order of left-most child node to right-most child node. There are two possible condition in this situation:

   (a) If the count of new numbers/nodes to be inserted is bigger than count of the previous child nodes, then again you must start from the previous left-most child node and continue to insert remaining new leaf nodes with order of left-most to right-most.

   (b) If the count of new numbers/nodes to be inserted is smaller than or equal to count of the previous child nodes, then you must normally distribute/insert the new leaf nodes to the previous child nodes with the same order mentioned above.

Part 2: Applying the Operations on Constructed Tree

In this part, you will perform several operations with respect to the second input file on final tree you constructed with respect to the first input file (See also Figure 2, middle). After each command you must use the latest modified tree for applying new operations on it.

1. **Deletion Command: Format:** "d" "number of node to be deleted"

   This command specifies that a specific node will be deleted on the tree with respect to the given number. If the given number doesn't exist or already deleted before, then just ignore the command and do not apply any operation on the tree. You must follow the rules below depending on the three possible situation:

   (a) If the node to be deleted is the root node, then you must delete the root node and you must set the new root node as the previous root node's left-most child. You

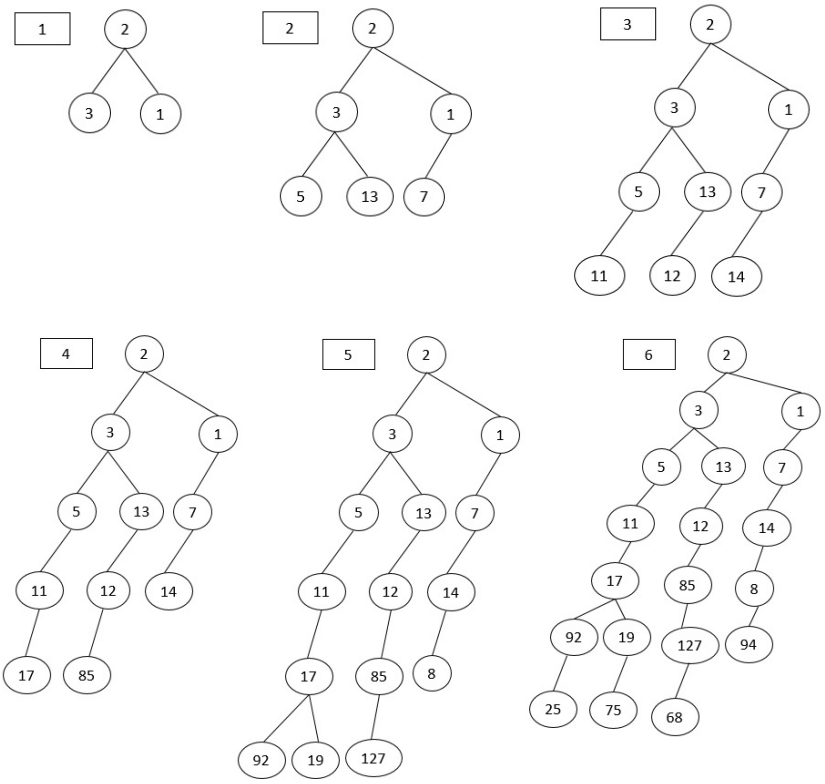

Figure 3: An example for construction of a tree with respect to the first input file in the Figure 2.

must also insert the previous root node's child nodes to the new root node as child nodes with keeping the left to right child node order for the both previous root node and the new root node (See Command 2 in Figure 4).

(b) If the node to be deleted is a middle node (that means it is both parent and child node), you must delete the middle node, then you must insert the deleted node's child nodes to the parent of the deleted node, again with keeping the left to right order for the both the deleted node and the parent of the deleted node. (See Command 1 in Figure 4 and also see Figure 5)

(c) If the node to be deleted is the leaf node, then simply delete this node with keeping the left to right order. (See Command 3 in Figure 4)

2. **List Command: Format:** "l" "number of node"

   In this command you must list the latest modified sub-tree from starting the node which's number is specified within the command. You must use **Depth-First Search, Preorder** traversal method while listing the tree. You must also write your list to the output file as a line (See Figure 2 right-most, see also Command 4-5 in Figure 4). If the given number doesn't exist or already deleted before, then just ignore the command and do not list the tree.

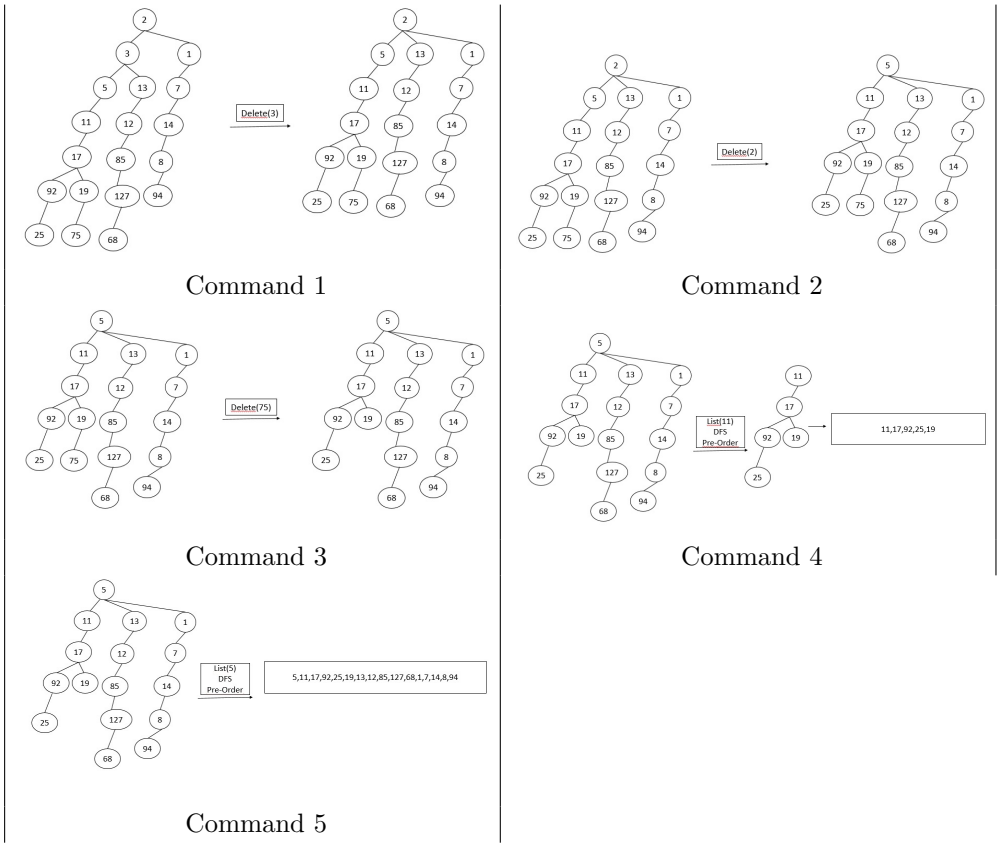

Figure 4: An example visual for applying operations on the constructed tree with respect to the second input file in the Figure 2.

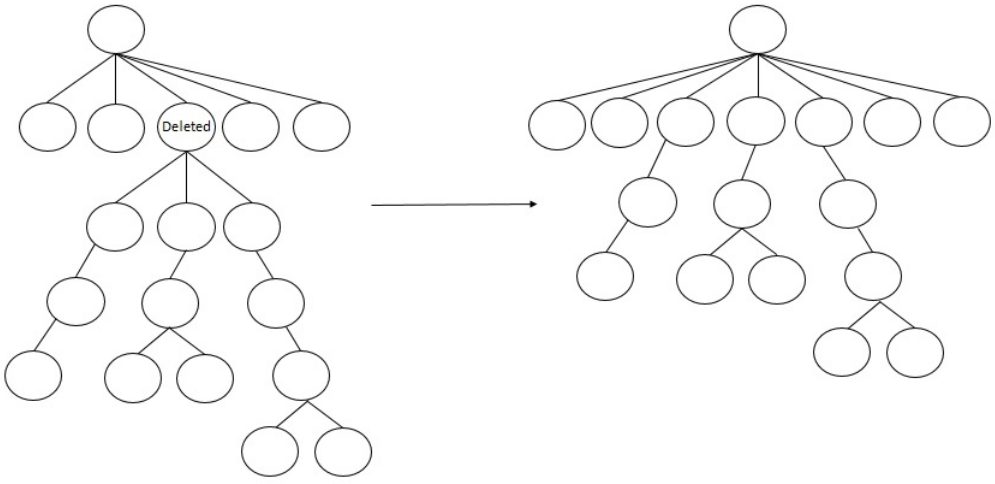

Figure 5: An example visual for the second situation for the deletion process

During run-time you must enter the first input file path and second input file path from the console respectively and your program must produce the related output file in it's working directory.

Report

1. Please explain your design, data structures, and algorithms.
2. Give necessary details without unnecessary clutter.
3. Do a Big-O notation algorithm analysis of your functions
4. Guide: ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.pdf

Important Issues and Notes

- Regardless of the length, use UNDERSTANDABLE names to your variables, classes, and Functions
- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Copying without citing will be considered as cheating. Citing does not make it yours; cited work will get 0 from the respective section.
- You can ask your questions via Piazza (`https://piazza.com/class/j7yfva1krme4gz?cid=15`) and you are supposed to be aware of everything discussed in Piazza.
- The submissions whose upload score is 0 will not be considered for evaluation.
- You will submit your work from `https://submit.cs.hacettepe.edu.tr/index.php` with the file hierarchy as below:
- **The experiment code will be tested on the dev machine. Your source code should be compiled with GCC. Otherwise your experiment will not be evaluated.**
- **Projects without a proper Makefile wont be graded.**

This file hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

→ <student id>
  → Report
    → report.pdf
  → Source
    → *.c
    → *.h
    → Makefile

This file hierarchy must be zipped before submitted (Not .rar , only .zip files are supported by the system).