Computer Engineering Department

TED University

CMPE 252 C Programming, Spring 2020

TERM PROJECT - BONUS PART

26.05.2020, Tuesday

DUE DATE: 16.06.2020, Tuesday, 23.59

Send your C code file via Moodle.

This is an individual work. No team work is allowed.

Similarity check will be applied to submitted codes.

BONUS QUESTION FOR TERM PROJECT

This part of the term project constitutes the BONUS part. It includes implementation of additional commands on the network tree which are:

- Moving a subtree under another subtree,
- Counting number of given node types under a subtree,
- Printing the tree / subtree in “Level Order”.

1 PROBLEM

1.1 MOVING A SUBTREE UNDER ANOTHER SUBTREE

In a dynamic network structure, a node can be disconnected from one location and connected to another location. This will be reflected on the constructed network tree by moving that node along with all the subtree to the given parent node. The “Move” operation should not change the Unique IDs of any nodes in the network.

The below table shows the command that is used to “move” a subtree of nodes to another parent node.

Table 1 : MOVE Command

MOVE <*node*>[<*ID*>] <*parent*>[<*ID*>]	This command removes the given <node>[<ID>] from its current parent node and connects it to the new given <parent>[<ID>]. The connection should be performed from the first empty child link of the new parent node from left to right.

Please refer to below example which demonstrates the move operation where Figure 1 shows the tree structure before the move operation and Figure 2 shows the network tree after the move operation.

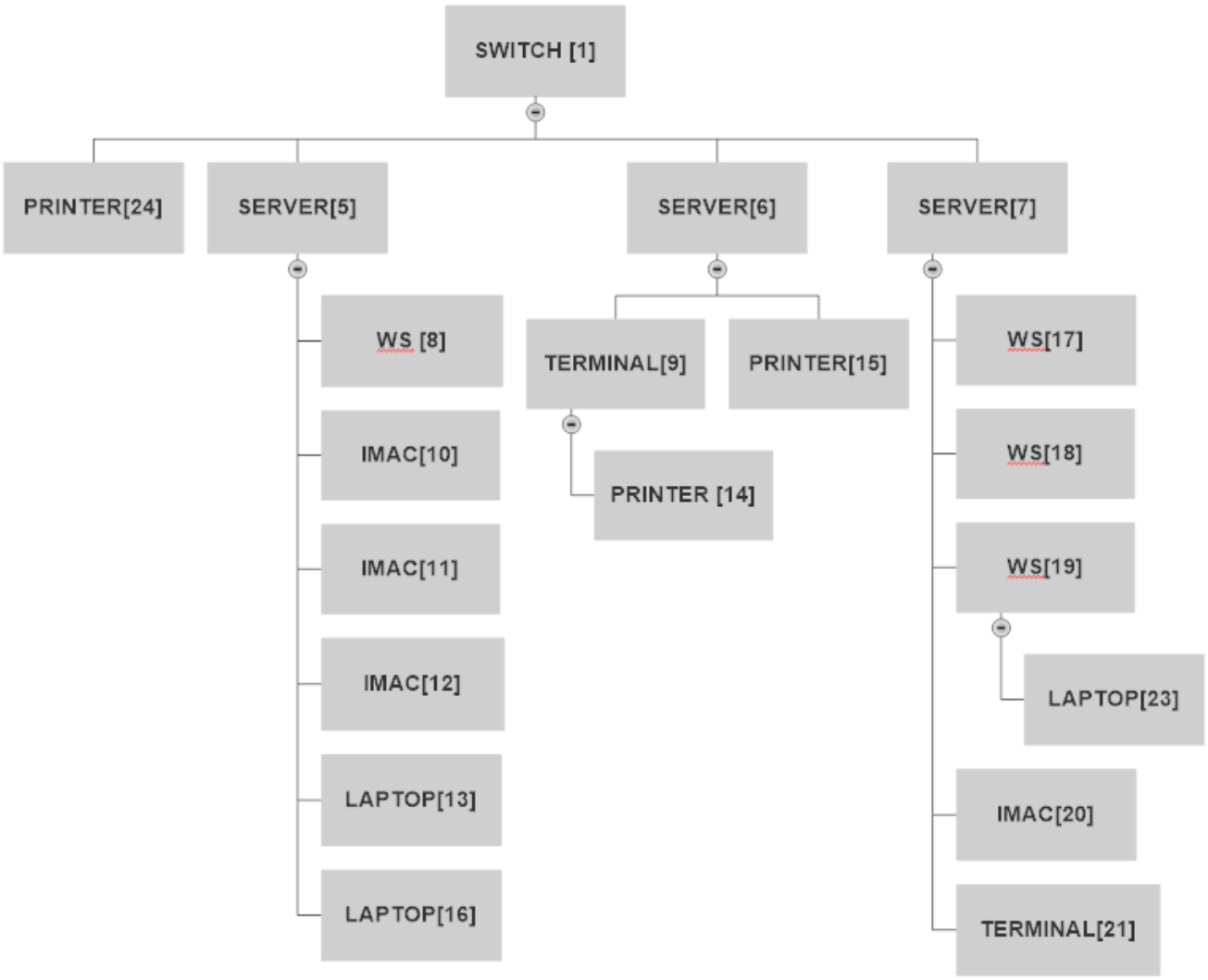

Figure 1: Network Tree Before MOVE Operation

When, the below sample input script command line is used to move a subtree in the network tree whose diagram is depicted in Figure *1*, the new network tree structure will be as depicted in Figure 2:

```
MOVE SERVER[6] PRINTER[24]
```

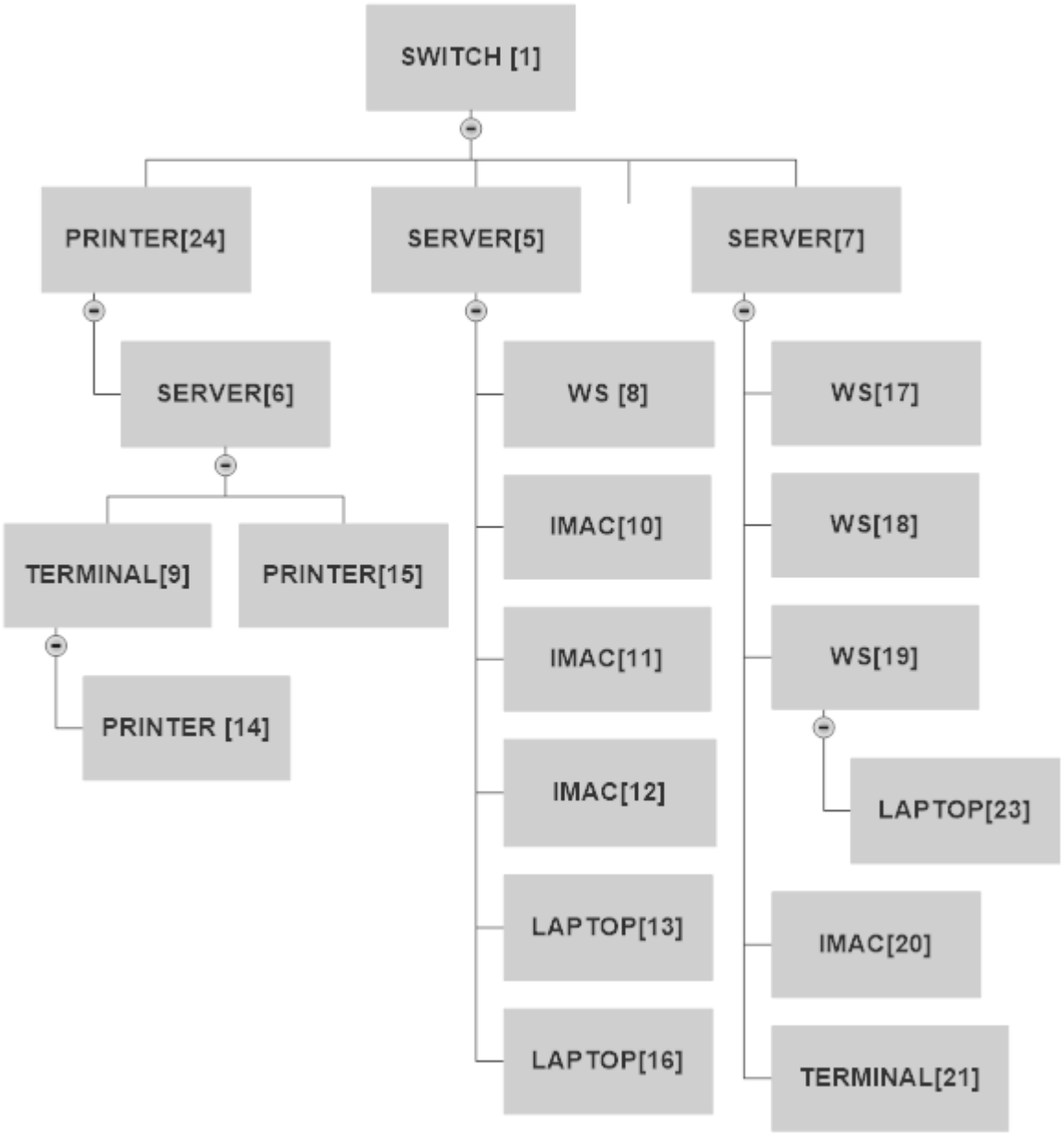

Figure 2 : Updated Tree of Sample Network After MOVE Operation.

1.2 COUNTING NODE TYPES IN THE TREE/SUBTREE

Network operators commonly need to count number of nodes of a specific type under the network or a sub-network. The below table shows the command that is used to "count" nodes of a specific type under the network tree or a subtree:

Table 2 : COUNT Command

COUNT <*nodetype*> <*parent*>[<*ID*>]	This command counts the number of nodes under the subtree provided by the <*parent*>[<*ID*>]. If it is the root node, then all the

	network is traversed for counting the given node type.

For example, assume, the below sample input script command lines are used to for several count operations regarding the network tree given in Figure 2.

```
COUNT WS SWITCH[1]
COUNT IMAC SWITCH[1]
COUNT LAPTOP SWITCH[1]
COUNT WS SERVER[5]
COUNT WS SERVER[7]
COUNT LAPTOP SERVER[7]
COUNT TERMINAL PRINTER[24]
COUNT PRINTER SWITCH[1]
COUNT PRINTER SERVER[7]
```

Below outputs should be provided, according to above commands:

```
4
4
3
1
3
1
1
3
0
```

1.3 PRINTING THE NETWORK TREE IN LEVEL ORDER

Due to the dynamic structure of the network, we often need to understand the current structure of the network.

Printing the network structure in “level order” (also called “breadth first search”) is also needed which provides network administrator the levels and deepness of the network.

The below table shows the command that is used to print the tree or the given subtree in “LEVEL ORDER”.

Table 3 : PRINT_LEVEL Command

PRINT_LEVEL *<node>[<ID>]*	This command prints the tree or subtree starting with the given node in LEVEL ORDER.

For example, “PRINT_LEVEL SWITCH[1]” and “PRINT_LEVEL SERVER[7]” commands for the tree given in Figure 2 will print to the output file the below output lines of strings corresponding to LEVEL ORDER visited nodes:

Output of “PRINT_LEVEL SWITCH[1]” command	**Output of “PRINT_LEVEL SERVER[7]” command**
SWITCH[1] PRINTER[24] SERVER[5] SERVER[7] SERVER[6] WS[8] IMAC[10] IMAC[11] IMAC[12] LAPTOP[13] LAPTOP[16] WS[17] WS[18] WS[19] IMAC[20] TERMINAL[21] TERMINAL[9] PRINTER[15] LAPTOP[23] PRINTER[14]	SERVER[7] WS[17] WS[18] WS[19] IMAC[20] TERMINAL[21] LAPTOP[23]

2 REQUIREMENTS

1. **Your bonus source code shall be written into 3 files (main.c, tree.c/.h) developed in the main part.**
2. **The bonus part of the program shall be executed at the same time with the main part as given below:**

>program.exe *<InputCommandsScriptFilename>* *<OutputFilename>* *<BonusInputScriptFilename>* *<BonusOutputFilename>*

```
>project.exe TreeCommands.txt Out.txt BonusCommands.txt BonusOut.txt
```

!! Bonus part functions shall process the final network structure after the main part functions are completed. The given commands in <BonusInputScriptFilename> script file will be processed over the network and the outputs are written to <BonusOutputFilename> output file.

The bonus part command line arguments are described as below:

< *BonusInputScriptFilename* > : The bonus part input commands script filename which includes commands given in this document (MOVE, COUNT and PRINT_LEVEL commands) in any order.

Same rules as given in the main part apply to the script.

Below is a valid input command script file opened in a text editor:

```
MOVE SERVER[6] PRINTER[24]
COUNT WS SWITCH[1]
COUNT IMAC SWITCH[1]
COUNT LAPTOP SWITCH[1]
COUNT WS SERVER[5]
COUNT WS SERVER[7]
COUNT LAPTOP SERVER[7]
COUNT TERMINAL PRINTER[24]
COUNT PRINTER SWITCH[1]
COUNT PRINTER SERVER[7]
PRINT_LEVEL SWITCH[1]
MOVE WS[19] PRINTER[14]
PRINT_LEVEL SWITCH[1]
```

< *BonusOutputFilename* > : The output file name which includes the output of the COUNT and PRINT_LEVEL statements processed in the order of appearance in the input bonus command script file.

3. **Along with this TERM PROJECT description, you will be provided;**
   - 5 sample debug case input / output files:

(“Inp1.txt Out1.txt”, “Inp2.txt Out2.txt”, “Inp3.txt Out3.txt”, “Inp4.txt Out4.txt”, “Inp5.txt Out5.txt”)

- 5 sample bonus part debug case input / output files : (“BonusInp1.txt BonusOut1.txt”, “BonusInp2.txt BonusOut2.txt”, “BonusInp3.txt BonusOut3.txt”, “BonusInp4.txt BonusOut4.txt”, “BonusInp5.txt BonusOut5.txt”)

which are generated for your own debugging purposes. Please make sure that your output files are “exactly the same” as these output files corresponding to below sample execution of our ground truth software so that you will be sure that your solution and the output format is correct.

```
TermProject.exe Inp1.txt Out1.txt BonusInp1.txt BonusOut1.txt
TermProject.exe Inp2.txt Out2.txt BonusInp2.txt BonusOut2.txt
TermProject.exe Inp3.txt Out3.txt BonusInp3.txt BonusOut3.txt
TermProject.exe Inp4.txt Out4.txt BonusInp4.txt BonusOut4.txt
TermProject.exe Inp5.txt Out5.txt BonusInp5.txt BonusOut5.txt
```

3 GRADING

(7 pts)

For grading, your executables will be executed using ;

- **5 debug test case parameters given along with the project description,**
- **5 more other interesting test cases that will not be provided to you, by using different input scripts which has no number of commands limitation requirement,**

and exactly the same output with the ground truth solution will be expected from your software.