**Problem 2:**

Description of the problem:

This program recursively generates a process tree (inside the rNode function) and forks when a tree node with children is found. The child process then creates and waits on the child child process to terminate so that the process tree is gradually terminated as well. If there is no children, the recursive functions sleeps for 3 seconds. In addition, each tree\_node contains its children\_no, name, left child, and right child.

How to run the program:

1. assignment2.c contains the main functions and node.c contains the tree\_node structs. Both are required to run the program.
2. input2.txt contains the given example scheme 1 shown in problem 2. The program requires identical character formatting of the given example. Changes such as removing spaces, numbers, or adding unnecessary characters will not run.
3. The program accepts exactly 1 argument of .txt for the input file.

*example code to run on gcc:*

gcc -o test assignment2.c

./test input2.txt

Question:

*What is the size of the process tree you selected is and why?*

The size of the process tree is dynamic due to the possibility of different inputs. The process tree’s size is limited to the amount of tree\_nodes it spawns due to creating and waiting on a child process for node creation. Each tree\_node is allocated the size of its struct data types as shown: struct tree\_node \*temp = (struct tree\_node\*)malloc(sizeof(struct tree\_node)). input2.txt was selected due to it being the given example scheme.

*What is the order of appearance of the start and termination messages from the processes and why?*

The order of appearance/termination of the tree is depth-first since the creation/printing function is recursive. Recursion requires the process tree to iterate down the depths until it reaches a leaf node, then it returns to iterate down until it hits the other side’s leaf node.

(output is attached below)

Output (also attached in assignment2output.c):

