Name: Rounak Sarkar
Roll No.: 21CS8118
Assignment: Q-A4

A4) Write a suitable code to create a process hierarchy with parent process be the root and the remaining child processes created in such a way they form a hierarchy of a full binary tree with depth ‘n’. You should take ‘n’ as input.

Each process including the parent and child processes does the following:

(a) display the pid & ppid once in the terminal and also store the values with timestamp in a common file (log.txt)

(b) sleep for 1 minute (this is just to allow the process tree to be visualized)

Hint: The parent will create the file log.txt and all its child processes can get access it. Get a snapshot of the process tree using forest option in ps command (check the pdf material provided). Provide a snapshot of the file log.txt

Code:

```
#include <iostream>
#include <fstream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <ctime>
#include <cmath>

using namespace std;

void log_pid_ppid(int pid, int ppid, ofstream& log_file) {
    time_t now = time(0);
    char* dt = ctime(&now);
    log_file << "PID: " << pid << ", PPID: " << ppid << ", Timestamp: " << dt;
    cout << "PID: " << pid << ", PPID: " << ppid << endl;
}

void create_tree(int depth, int pid, ofstream& log_file) {

    if (depth == 0){
        log_pid_ppid(getpid(), pid, log_file);
        sleep(20);
        return;
    }

    pid_t left_child_pid, right_child_pid;
    left_child_pid = fork();
    if (left_child_pid == -1) {
```

```
        cerr << "Error: fork failed" << endl;
        return;
    } else if (left_child_pid == 0) {
        // left child process
        log_pid_ppid(getpid(), getppid(), log_file);
        create_tree(depth - 1, getpid(), log_file);
        return;
    } else {
        // parent process
        right_child_pid = fork();
        if (right_child_pid == -1) {
            cerr << "Error: fork failed" << endl;
            return;
        } else if (right_child_pid == 0) {
            // right child process
            log_pid_ppid(getpid(), getppid(), log_file);
            create_tree(depth - 1, getpid(), log_file);
            return;
        } else {
            // parent process
            log_pid_ppid(getpid(), pid, log_file);
            wait(NULL);
            wait(NULL);
        }
    }
}

int main() {
    int n;
    cout << "Enter the depth of the full binary tree: ";
    cin >> n;

    int num_nodes = pow(2, n) - 1;
    cout<< "The number of nodes will be: " << num_nodes << endl;

    ofstream log_file("log1.txt");
    if (!log_file.is_open()) {
        cerr << "Error: could not open log file" << endl;
        return 1;
    }

    create_tree(n, getpid(), log_file);

    log_file.close();

    return 0;
}
```