CS 3113 Intro to Operating Systems
Name and ID Ruben Osornio 113537774
Homework #3
Instructions:
1) To complete assignment one, you need to read Chapters 1, 3 and 4 of the textbook.
2) HW must be submitted on typed pdf or word document.
You must do your work on your own.
Q1. Using Amdahl's Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores. (15 points)

A. 1.43
B. 1.82

Q2. A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between start-up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread). (20 points)

a. How many threads will you create to perform the input and output? Explain.

Create 1 thread total for input and output The input and output are sequential and limited by the I/O system, so adding more threads would not improve performance. One thread for each operation ensures no thread contention over file access.

b. How many threads will you create for the CPU-intensive portion of the application? Explain.

The system has 4 processors, so creating 4 threads will allow each core to work in parallel, maximizing the application's performance.

Q3. Consider the following code segment: (15 points)

```
pid t pid;
pid = fork();
if (pid == 0) { /* child process */
fork();
thread create( . . .); /* for the purpose of this problem, you can ignore the lack
of arguments to the function */
}
fork();
```

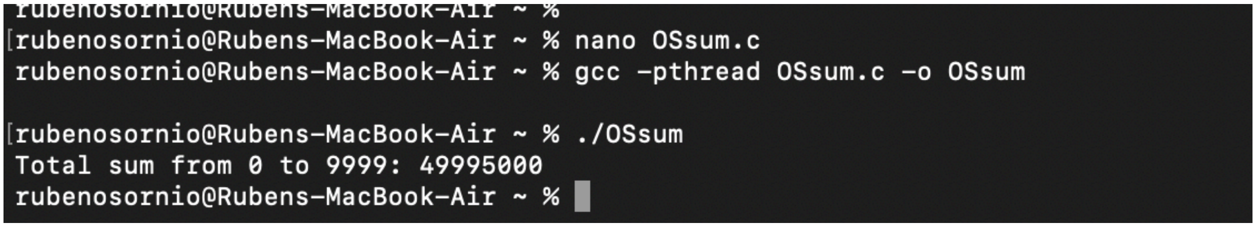
a. How many unique processes are created?

First it starts with 1 process.
First fork() creates 1 new process (now 2 processes).
Inside the child, a second fork() creates 1 more process (now 3 processes).
The third fork() is called by all 3 processes, creating 3 more processes.
So 6 unique processes are made.

b. How many unique threads are created?

The thread_create() is only called in the child process after the first fork(), creating 1 thread.
Further fork() calls do not create additional threads.
So 1 unique thread is created

Q4. Pthread programming: writing a program to join on ten threads for calculating 0-9999. Each thread calculates the sum of 1000 numbers. Please attach screenshots of your execution results
below. You also need to submit your code (along with a readme file) separately. (50 points)
All files (MUST INCLUDE: source codes, a readme file, and homework 3) should be zipped together.



```
 rubenosornio@Rubens-MacBook-Air ~ %
[rubenosornio@Rubens-MacBook-Air ~ % nano OSsum.c
 rubenosornio@Rubens-MacBook-Air ~ % gcc -pthread OSsum.c -o OSsum

[rubenosornio@Rubens-MacBook-Air ~ % ./OSsum
 Total sum from 0 to 9999: 49995000
 rubenosornio@Rubens-MacBook-Air ~ %
```