Aki Gao (xag9bb)
Homework 2
CS 4414 – Operating Systems
Fall 2015

1.
   a) One thread is needed for both input and output. Because input and output operations to the system are I/O bound, having multiple threads will not improve the performance of the program.

   b) Four threads will be created to optimize the performance because the program is CPU-bound between startup and termination, meaning that the time for completion of the task is determined by the speed of the CPU. Therefore, utilizing all four processors would best improve the performance of the program.

2.

   a) Four unique process are created – one for the original parent process, one for the child process on the first fork call, one for the child process created by the first child process, and another for the second fork call in the parent process.

   b) Three unique threads will be created – one for the original process, one in the thread create function call in the first child process created, and another in the thread create function call for child process of the first child process.

3.
   Line C – CHILD: value = 5
   Line P – PARENT: value = 0

4.
   a) The function first declares a global counter that is initialized to 0. The it defines a struct list that contains a pointer to another struct list and a double value. This struct is then reassigned as a pointer list. The function count_positives(…) takes in a pointer to struct list as a parameter, and counts the number of positive values in the list by iterating through pointer next, which points to another struct list.

   b) Thread A would not increment the global value at at all, while thread B will increment the global value by 1. The two threads will run without any conflict.

   c) The use of two parallel threads, if the code is not optimized for multithreading, can lead to race conditions in which results from one thread can overwrite results from another thread if they both operate on the same global variable(s).

5. If count_positives is optimized in such a manner, both thread A and B will modify the global variable. If both threads are executed concurrently, then race conditions will occur, and the final value of global_positives will vary by execution of the program.