CIS 452 03 - Lab 04

**Start up a Terminal session, which provides you with the UNIX command-line interface. Compile and run sampleProgramOne. You will need to link in the pthreads library (i.e. you must compile with the -pthread compiler option). Observe the results. Try inserting a 2-second sleep() into the main() function (after thread creation); compile and re-run.**

1. **Describe/explain your observations, i.e., what must have happened in the original, unmodified program? (4 points)**

When running the original program, the program prints “Running the program” and then quits. The program doesn’t print “Thread version of hello world” because the main thread quits before the newly created thread is able to execute the code. We can change this by adding a 2-second sleep system call into the main function, after the thread has been created. When we add this line in, we end up seeing the “Thread version of hello world” being printed after “Running the program”.

**Compile and run sampleProgramTwo. Run sampleProgramTwo multiple times.**

1. **What does sampleProgramTwo output? If you run it a repeated number of times does the output vary? Why? (6 points)**

sampleProgramTwo outputs the strings “Hello” and “Good Bye” multiple times, along with which string was passed and to what thread. When the program is finished, both threads return and the program prints. The output does vary when ran multiple times. The output varies because, even though we are waiting for thread1, thread2 is still able to execute.

**Insert a one-second sleep() at the beginning of the loop in the doGreeting() function. Compile and run the modified program.**

1. **Report your results again. Explain why they are different from the results seen in question 2. (4 points)**

sampleProgramTwo now outputs the creation of threads 1 & 2 followed by one another, and then prints the string “Hello” and “Good Bye” 10 times after each other. This is because the sleep(1) call in the doGreeting for loop gives each thread time to print their corresponding string before sleeping again and allowing the next thread to print their corresponding string.

1. **Compile the sample program and run it multiple times (you may see some variation between runs). Choose one particular sample run. Describe, trace, and explain the output of the program. (6 points)**

First, the two threads are created. As each thread is created, their assigned routine is doGreeting. After these two threads are created, the parent prints “Parent sees 5” because the sharedData variable that all of the threads and the parent have access to is 5 at the moment and has not been modified. Then, each child prints “Child receiving b initially sees 5” and “Child receiving a initially sees 5”, as the sharedData has not been modified yet. Next, the parent adds 1 to sharedData, making it 6, but nothing is printed yet due to the two pthreads sleeping for 1 second. Then, the child thread receiving a adds 1 to sharedData and prints “Child receiving a now sees 7 as the child just added 1 to the sharedData that was just 6. Next, the other child thread also adds 1 to the shared data and prints “Child receiving b now sees 8”, and sharedData is also now 8. While this is going on, the parent is joining both of the threads and storing them in joinStatus1 and joinStatus2 and doing some error checking with those. Finally, the parent prints “Parent sees 8” after the sharedData has been manipulated by the parent thread and both pthreads.

1. **Explain in your own words how the thread-specific (not shared) data is communicated to the child threads. (4 points)**

The data that is not shared is communicated to the child threads through the argument parameter of the pthread\_create method used to create the pthreads.