**Assignment 3 Report**

**Gabriel Antonio Alvim D’Arco**

**Problem 1: The Birthday Presents Party**

**Proof of Correctness:**

The program starts by creating 4 threads that represent the 4 different Minotaur Servants and it starts their processes. Each thread will then start its run method, which will generate a random number which will represent which of the three tasks that thread will try to do. The first one is when a servant tries to take a present from the box of presents and place it into the linked list called Present. The second tries to take the first present from the linked list out of the chain and give it a thank you card. The third task is to search for a random tagged present and see if it is present in the linked list. Each thread always prints what it is doing at the moment. After all presents have been taken from the box and at some point placed in the linked list, if at some point a servant tries to place a new present in the chain, it prints out saying that there aren’t any presents left. The program is guaranteed to run until all presents are taken out of the chain and given a thank you card because of the variable “giftsThanked”, which keeps track of how many presents have been given a thank you card.

**Efficiency:**

The efficiency of the program is as good as it can be with a basic linked list with linear search and insertion time. It could definitely be improved if using a more advanced data structure, like skip lists for example.

**Experimental Evaluation:**

As far as testing for the program, I ran the code at least 20 times, and in all executions it finished at around the same time. Additionally, it never threw any errors and always finished after all presents were “Thanked”, guaranteeing that the main objective was completed.

**Problem 1: Atmospheric Temperature Reading Module**

**Proof of Correctness:**

The program starts by immediately spawning and starting all 8 threads that represent each temperature sensor. After that, the threads will all generate a random temperature from -100F to 70F to simulate the temperature reading. After that, the program checks to see if the temperature measured is higher than any temperature stored in the 5 highest temperatures or lower than any of the 5 lowest temperatures that are stored. If so, it replaces that temperature. After time passes, the hourly report is printed, showing the top 5 lowest and highest temperatures recorded during the measurement time. After that, it starts the process over again after wiping the previously recorded highest and lowest temperatures. The program will only terminate if the user forces it.  
  
Disclaimer: The requirement for the assignment was that each measurement for all threads would happen every 1 minute and the report would be made every hour. However, to make testing easier and to help the grading process as well, I made it so that measurements would happen every second, and the report would be made every minute. That way I could easily test the program without waiting an hour.

**Efficiency:**

The program is very efficient. Obviously run time would depend on exactly how much time you would want each thread to wait to take a measurement. As far as storage goes, all threads store the highest and lowest temperatures in the same two Atomic Integer Array variables, and it never causes issues.

**Experimental Evaluation:**

For testing, I ran the program 5 times, and in all executions I let it create three reports. The program worked flawlessly and managed to create the reports correctly without any issues.