Damian Franco

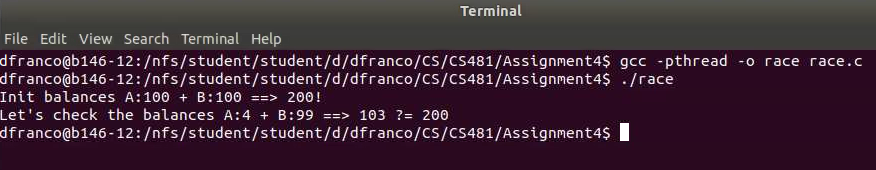
Meiling Traeger

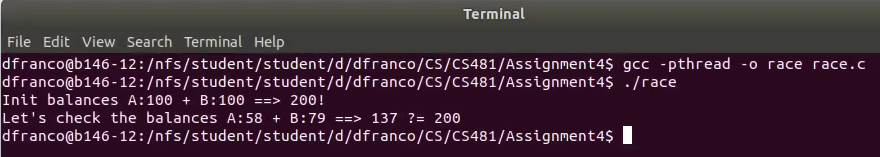
CS 481-003

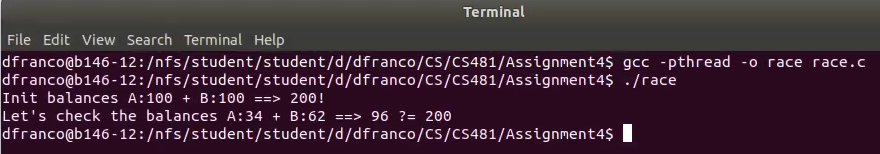
Programming Assignment #4

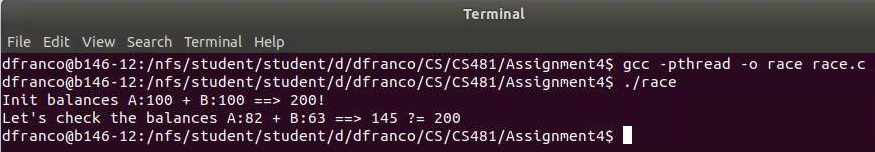
1. For number 1, we were asked to run the C program (*race.c*) about 20-40 times and observe the output. We observed that the outputs are very random with no real consistency to them. Randomization of the balances occurs within the *MakeTransactions()* function call. The function call will assign the two bank values to a random integer by utilizing the C function *rand()* and by creating and manipulating temporary integers. Most of the output is not out of the ordinary, but it seems to be erroneous because the comparison statement within the last printf “*=? 200*” implies that both integers should have the sum of 200, but it does not. If any errors are found within this program, they mostly lie with the way the program operates. Possible race conditions may be occurring here which can cause errors in the scheduling and timing of the current threads and even possibly lead to errors within the program's behavior. We believe race conditions occur in this program because the shared data seems to be getting accessed at the same time and leads to some mysterious outcomes when assigning new values to the shared memory. To counteract that, a critical section must be reached, which is what we will be trying to achieve in question 2.

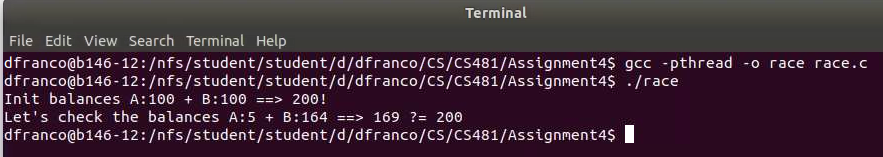
**Output Screenshots:**

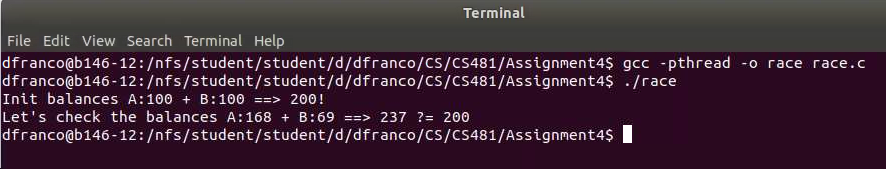
*Output after running* **1** *time:*

*Output after running* **5** *times:*

*Output after running* **10** *times:*

*Output after running* **20** *times:*

*Output after running* **30** *times:*

*Output after running* **40** *times:*

1. For number 2…
2. For number 3…
3. For number 4…