# Pair Programming Equitable Participation & Honesty Affidavit

We the undersigned promise that we have in good faith attempted to follow the principles of pair programming. Although we were free to discuss ideas with others, the implementation is our own. We have shared a common workspace and taken turns at the keyboard for the majority of the work that we are submitting. Furthermore, any non programming portions of the assignment were done independently. We recognize that should this not be the case, we will be subject to penalties as outlined in the course syllabus.

Carlos Gamino, 819230978 04/16/2020

Pair Programmer 1 (print & sign your name, then date it)  
  
  
  
Tan Truong, 821006778 04/16/2020

Pair Programmer 2 (print & sign your name, then date it)

Carlos Gamino Reyes Tan Truong

misc 0230 misc 0308

819230978 821006778

**Question 1:**

We could not figure out the answer to this question, even with the help of the tutor.

**Question 2:**

main() {

int x;

Semaphore s;

for (j = 0; j < NumberThreads; j++) {

start new thread executing foobar(&x);

}

}

foobar(int \*x) {

SemaphoreWait(s) // enter critical region

y = 2 \* (\*x);

print y;

z = 2 \* y;

\*x = \*x / 2;

SemaphoreSignal(s) // leave critical regions

}

**Question 3:**

The synchronization will always function as intended for the code.

Assuming:

semaphore barrier = NUM\_WORKERS = 0;

We have the Up-function increase semaphore by 1, and the Down function decrease semaphore by 1.

Thus, the line of code print "all done" is only execution after all the workers threads have done.

Besides, the synchronization won’t function as intended for the code if semaphore barrier is equal 0 and is not equal with NUM\_WORKERS.

If that happens, the semaphore will be locked and can’t continue to run all the threads.