AKM Jahangir Majumder, USC Upstate Fall 2019

Student Name:

CSCI U511 – Operating Systems

Homework-3, Weight: 40 points

Due on Tuesday, Oct 1, 2019 at the beginning of the lecture (Hard Copy)

Note: You need to include your calculation and details to receive full credit!

Please remember to write your name on your answer sheet.

Q1. For the following program, when thread_join returns for thread i, what is thread i's thread state? What is the minimum and maximum number of times that the main thread enters the WAITING state?

```
#define NTHREADS 10

thread_t threads[NTHREADS];

main() {

    for (i = 0; i < NTHREADS; i++)  thread_create(&threads[i],
&go, i);

    for (i = 0; i < NTHREADS; i++) {

        exitValue = thread_join(threads[i]);

        printf("Thread %d returned with %ld\n", i, exitValue);

    }

    printf("Main thread done.\n");

    return 0;

}

void go (int n) {

    printf("Hello from thread %d\n", n);

    thread_exit(100 + n);

    // REACHED?

   }
```

Q2. Show that solution 3 (discussed in lecture 8) to the Too Much Milk problem is safe—that it guarantees that at most one roommate buys milk.

Q3. Suppose that you mistakenly create an automatic (local) variable *v* in one thread *t1* and pass a pointer to *v* to another thread *t2*. Is it possible that a write by *t1* to some variable other than *v* will change the state of *v* as observed by *t2*? If so, explain how this can happen and give an example. If not, explain why not.

Student Name:

Q4. You have been hired by a company to do climate modelling of oceans. The inner loop of the program matches atoms of different types as they form molecules. In an excessive reliance on threads, each atom is represented by a thread.

Your task is to write code to form water out of two hydrogen threads and one oxygen thread ($H2O$). You are to write the two procedures: **HArrives()** and **OArrives()**. A water molecule forms when two H threads are present and one O thread; otherwise, the atoms must wait. Once all three are present, one of the threads calls **MakeWater()**, and only then, all three depart.

You must use locks and Mesa-style condition variables to implement your solutions. Obviously, an atom that arrives after the molecule is made must wait for a different group of atoms to be present. There should be no busy-waiting and you should correctly handle spurious wakeups. There must also be no useless waiting: atoms should not wait if there is a sufficient number of each type.