CS 3370 – Program 6

**Producer-Consumer Processing**

This short program extends the example in file *wait6.cpp* by adding a second queue and some post-processing threads.

Modify *wait6.cpp* so that the consumers add their numbers to a second queue. Create exactly 10 threads that read the second queue and group the numbers by their modulus base 10. Each of the 10 grouper threads will remove only those numbers from the queue whose modulus corresponds to theirs. For example, grouper thread 0 will remove numbers whose last digit is 0 and write it to the file *bin0.txt*. Each grouper thread does likewise for its respective residue base 10. If the number at the front of the queue does not end with the proper digit, the thread leaves it in the queue for the appropriate thread to process.

When you are done, print out a report like the following (your numbers will vary):

chucksretinambp:3370 chuck$ ./a.out

Press Enter to quit...

Group 0 has 6972 numbers

Group 1 has 7116 numbers

Group 2 has 7206 numbers

Group 3 has 7153 numbers

Group 4 has 7184 numbers

Group 5 has 7255 numbers

Group 6 has 6987 numbers

Group 7 has 7236 numbers

Group 8 has 7062 numbers

Group 9 has 7262 numbers

Collect your 10 output text files and your source code, along with the execution output like what you see above, into a zip file for submission. (*Note*: on Visual Studio, you may have to press Enter twice. This program may run faster from the command line than in the IDE. Wait up to 2 minutes for it to finish.)

I’m going to give you the test main function I used (because that’s how I roll… sometimes :-):

int main() {

vector<thread> prods, cons;

array<thread,10> bins;

for (int i = 0; i < ProducerConsumer::nconss; ++i)

cons.push\_back(thread(&ProducerConsumer::consume));

for (int i = 0; i < ProducerConsumer::nprods; ++i)

prods.push\_back(thread(&ProducerConsumer::produce));

for (int i = 0; i < 10; ++i)

bins[i] = thread(&ProducerConsumer::group, i);

cout << "Press Enter to quit...";

cin.get();

ProducerConsumer::quit = true;

// Join all threads

for (auto &p: prods)

p.join();

for (auto &c: cons)

c.join();

for (auto &b: bins)

b.join();

ProducerConsumer::report();

}