Contents

Lab 10

STL Queues

Topics: Algorithm Efficiency, Searching

Group work: Group work is allowed for the labs, but each person must do their own coding and each person must turn in an assignment. Copying other peoples' code / code files is not allowed. Copied assignments will receive 0% for everybody involved.

Assignments must build: Assignments that don't build will automatically just be given a flat 50% score. I may still give feedback on what went wrong, and 50% is better than 0%, so turn in something - but ideally, make sure it builds.

For this lab, we are implementing a simple job queue that contains a list of factorials to compute. Since the program can only process one item at a time, it will use a queue to deal with everything on a first-come, first-served basis.

Use the STL queue documentation for function examples:
http://www.cplusplus.com/reference/queue/queue/

10.1 Program output

The program output will look like this:

Processing jobs

```
17 jobs remaining.  Calculating 1! ... Result is 1
16 jobs remaining.  Calculating 12! ... Result is 479001600
15 jobs remaining.  Calculating 3! ... Result is 6
14 jobs remaining.  Calculating 3! ... Result is 6
13 jobs remaining.  Calculating 10! ... Result is 3628800
12 jobs remaining.  Calculating 1! ... Result is 1
11 jobs remaining.  Calculating 3! ... Result is 6
10 jobs remaining.  Calculating 10! ... Result is 3628800
9 jobs remaining.  Calculating 4! ... Result is 24
8 jobs remaining.  Calculating 0! ... Result is 1
7 jobs remaining.  Calculating 12! ... Result is 479001600
6 jobs remaining.  Calculating 2! ... Result is 2
5 jobs remaining.  Calculating 2! ... Result is 2
4 jobs remaining.  Calculating 4! ... Result is 24
3 jobs remaining.  Calculating 1! ... Result is 1
2 jobs remaining.  Calculating 3! ... Result is 6
1 jobs remaining.  Calculating 1! ... Result is 1
0 jobs remaining.
```

10.2 Program code

You will be working with the `FactorialComputer.hpp` file and implementing the queue. You will need to update two functions.

1. `FactorialComputer` class declaration:
   Add a queue of ints as a member variable.

2. `CreateJobs` function:
   Generate a random number and push the number into the job queue.

3. `ProcessJobs` function:
   While the job queue is not empty, process the next item then pop it off the list. To process the item, call `Factorial_Rec`, passing in the `m_jobQueue.front()` item.

10.3 Turn in

Once completed, turn in your `FactorialComputer.cpp` file into Canvas.