HoldenG
Data Structures Homework 3

Queue practice

// This program simulates jobs which are of random assigned lengths
// and have random delays between their arrival. All time values are
// actual time values using <chrono> at sub-millisecond units.
//
// The Simulation object creates a vector of Virtual Machine (VM) objects, each
// one containing a vector of CPU objects.
//
// The simulation also creates a Run Queue object for jobs that need to be
// processed, and a Completed Queue object for jobs that are done. Both are
// custom Template Linked Lists that hold Job objects.
//
// **For each CPU of each VM, the following logic is performed:**
//
// - Test if a new job has "arrived" based on time comparison against
// delay. If it has:
// Create the job object, stamp time, place it into the Run Queue, and
// increase Run Queue length. Write information about the queue event to
// Queue Event CSV file (including the size of the queue) - for import
// into Excel to visualize time and queue size with scatterplot diagram.
// Also test to see if the new job is the smallest or biggest so far.
//
// - Test if the current CPU has a job and is done processing it (elapsed
// time on cpu is longer than assigned length). If so, stamp time and move
// job to Completed Queue. Also, if the final job was just moved to
// completed queue, set simulation Done flag to true.
//
// - Test if the current processor is ready and there is a job available
// in the Run Queue. If so, stamp time, move job from Run Queue to the CPU,
// and decrease Run Queue length. Write information about the queue event to
// CSV file (including the new size of the queue).
//
// - If the simulation is not done, move to next Machine/CPU

Program Files:

main.cpp	// Main routine
Sim.cpp	// Simulation Object
Sim.h	
MyList.h	// Templated Linked List
MyQueue.h	// Extends MyList
VM.cpp	// Virtual Machine Obj
VM.h	
CPU.cpp	// CPU Obj
CPU.h	
Job.cpp	// Job Object
Job.h	

Program Generated Output Files:

QueueSequence.csv	// The order jobs enter and exit Run queue
Done_Queue.txt	// Entire listing of Job objs in done order

Additional Files for Analysis and Visualization:

doneQueue300.xlsx	// Excel graph of 300 job sequence
doneQueue4000.xlsx	// Excel graph of 4000 job sequence

Requirements:

Calculate the following information about each CPU utilization:

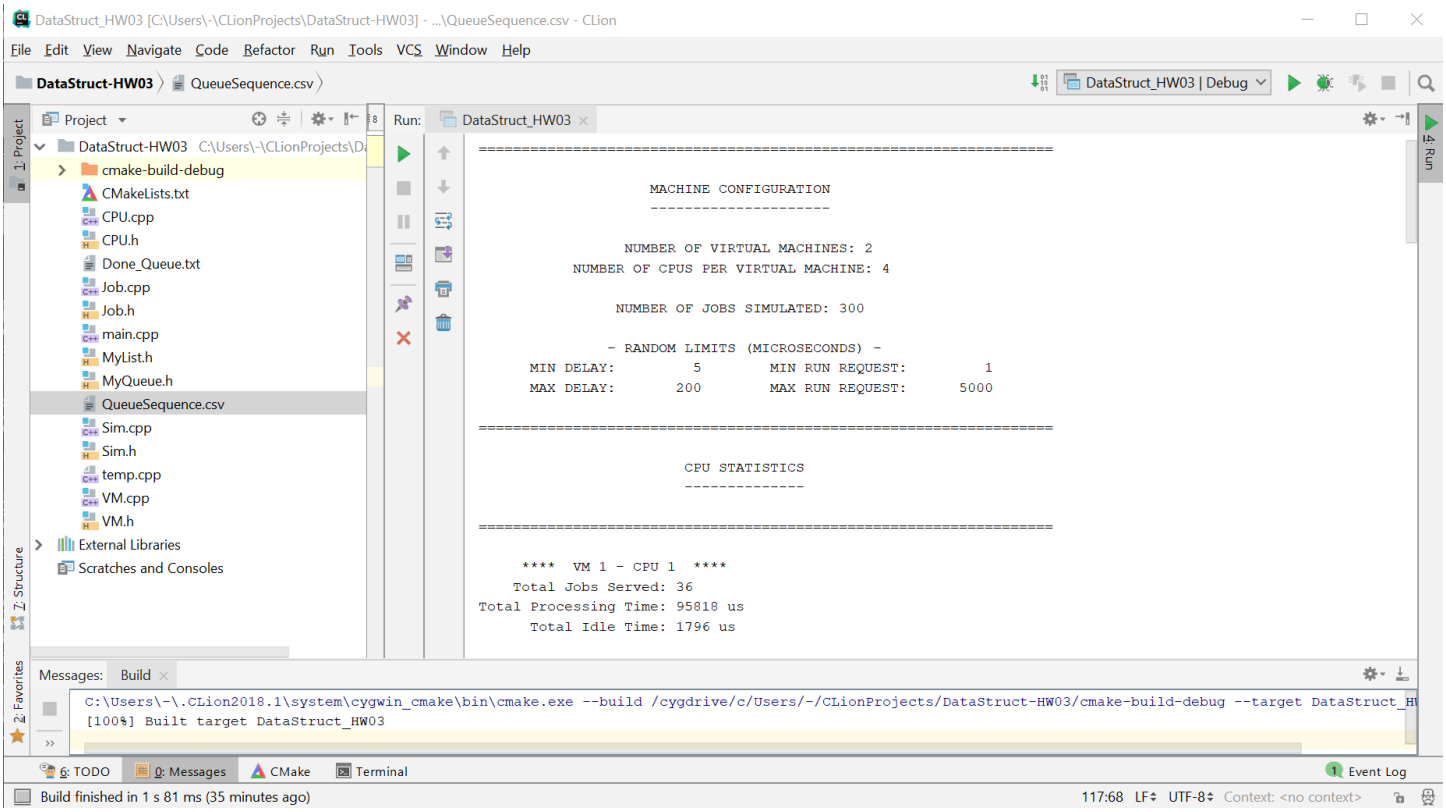
1. Total busy time
2. Total idle time
3. Number of jobs served by the CPU.

Use actual time as hours, minutes, and seconds instead of integer value to process each job and assign different jobs length.

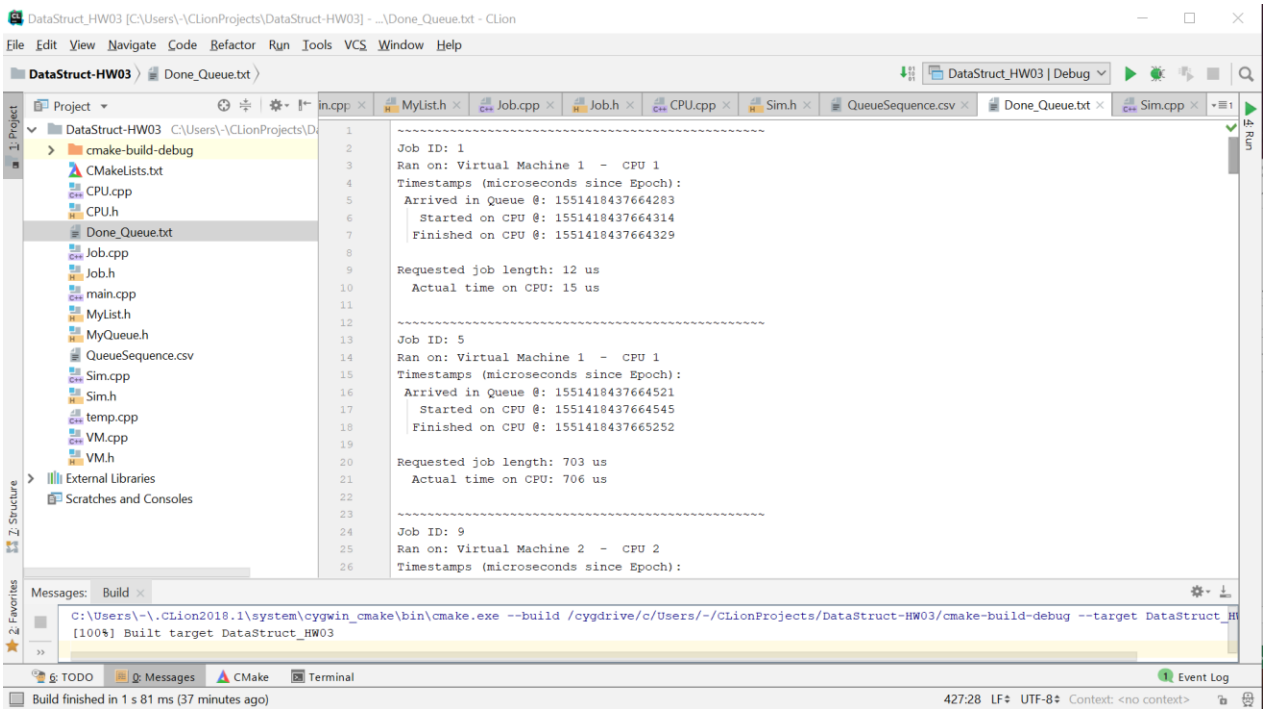
Also used csv file generated to import into Excel and graph queue size in relation to time.

Machine Configuration:
Summary of configuration including number of machines, CPUs per machine, number of jobs requested, and the limits used for random creation of job arrival delays and requested job lengths.

CPU Statistics:
Listing of each CPU on each Machine, Total jobs run, total busy time, and total idle time.



Completed job information written to Done_Queue.txt.
Includes ID number, which Machine and CPU it ran on, arrival times in run queue, CPU, & done queue, random length assigned, and actual run time.

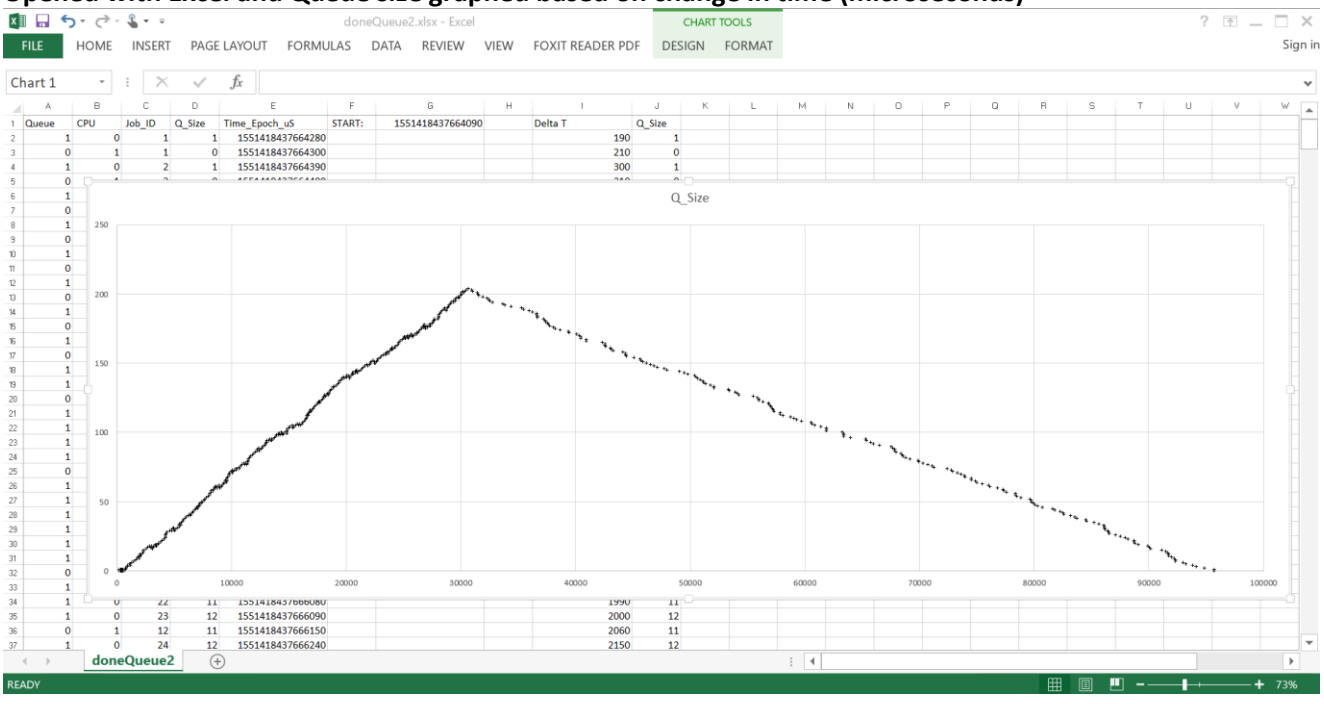


Done Queue sequence of events written to QueueSequence.csv :

```
toQueue, toCPU, Job_ID, Q_Size, Time_Epoch_uS, START:, 1551418437664094
1, 0, 1, 1, 1551418437664286
0, 1, 1, 0, 1551418437664305
1, 0, 2, 1, 1551418437664392
0, 1, 2, 0, 1551418437664401
1, 0, 3, 1, 1551418437664469
0, 1, 3, 0, 1551418437664480
1, 0, 4, 1, 1551418437664504
0, 1, 4, 0, 1551418437664513
```

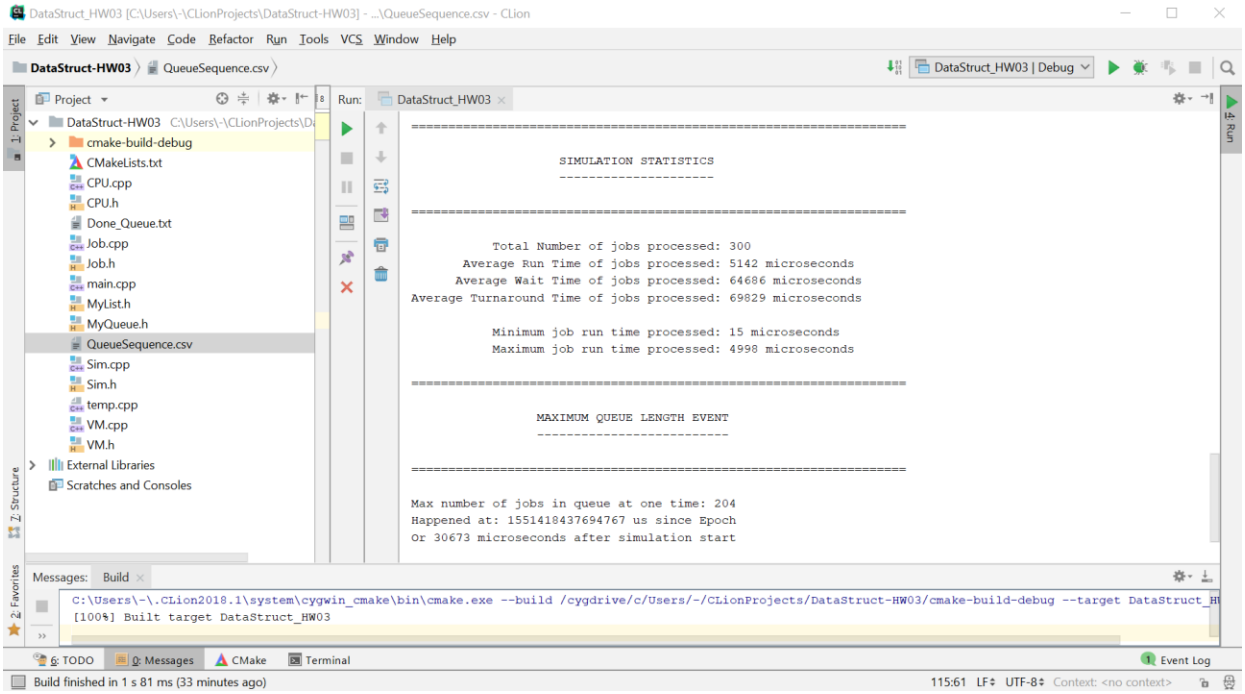
...

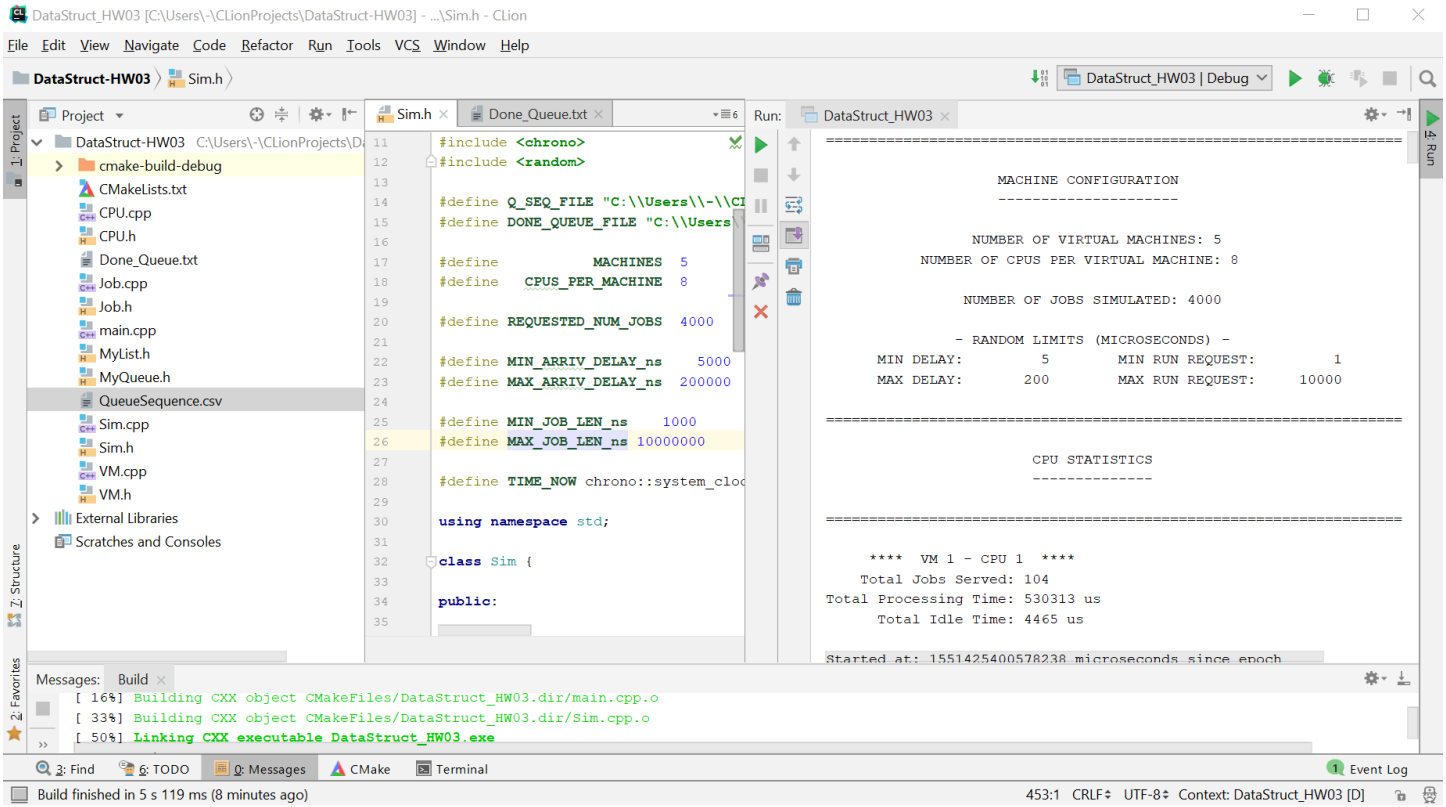
Opened with Excel and Queue size graphed based on change in time (microseconds)



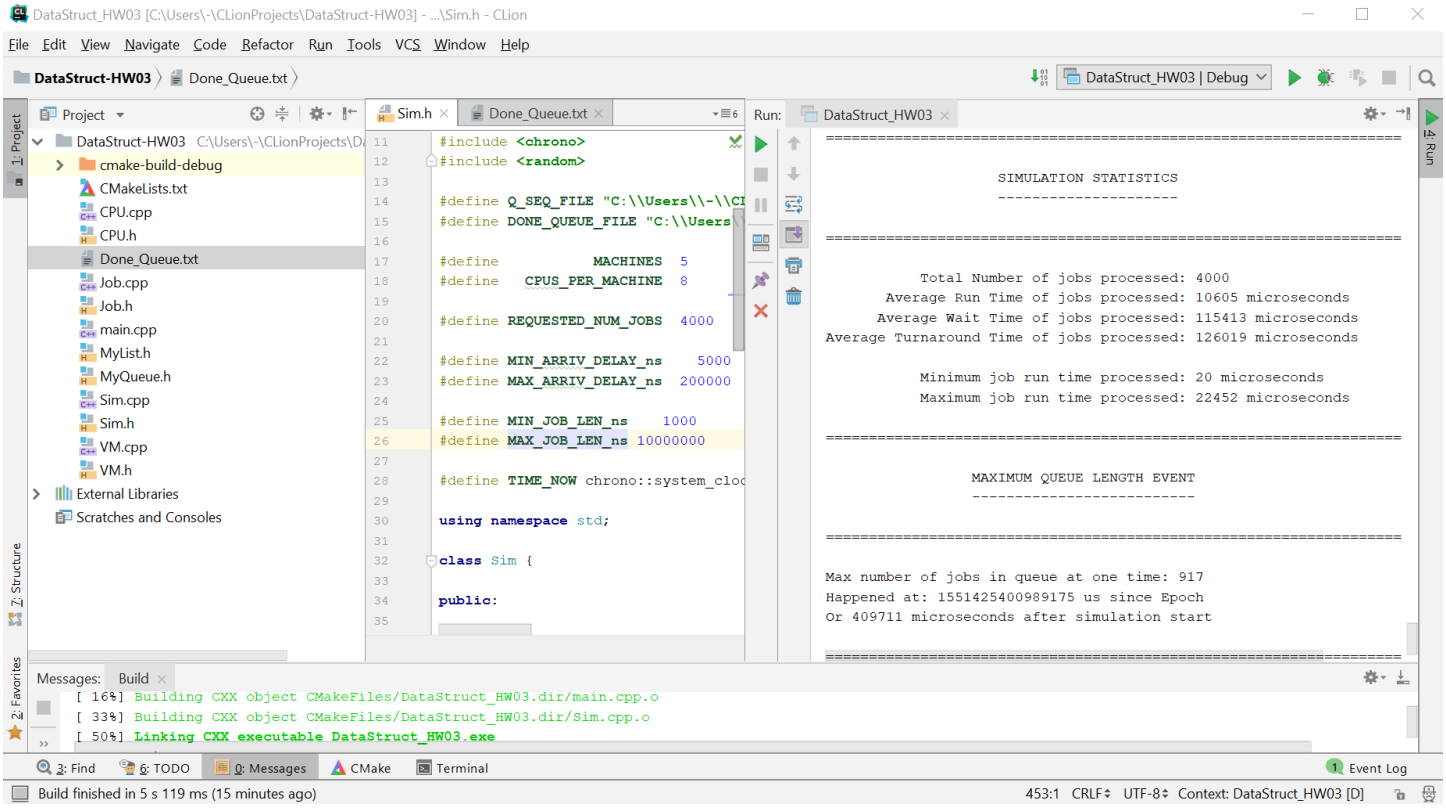
Overall simulation statistics: number of jobs run, average run time, average wait time, average turnaround time, minimum run time, and maximum run time.

Maximum number of jobs in queue at once and time it occurred. Maximum size and time match the information in the graph.

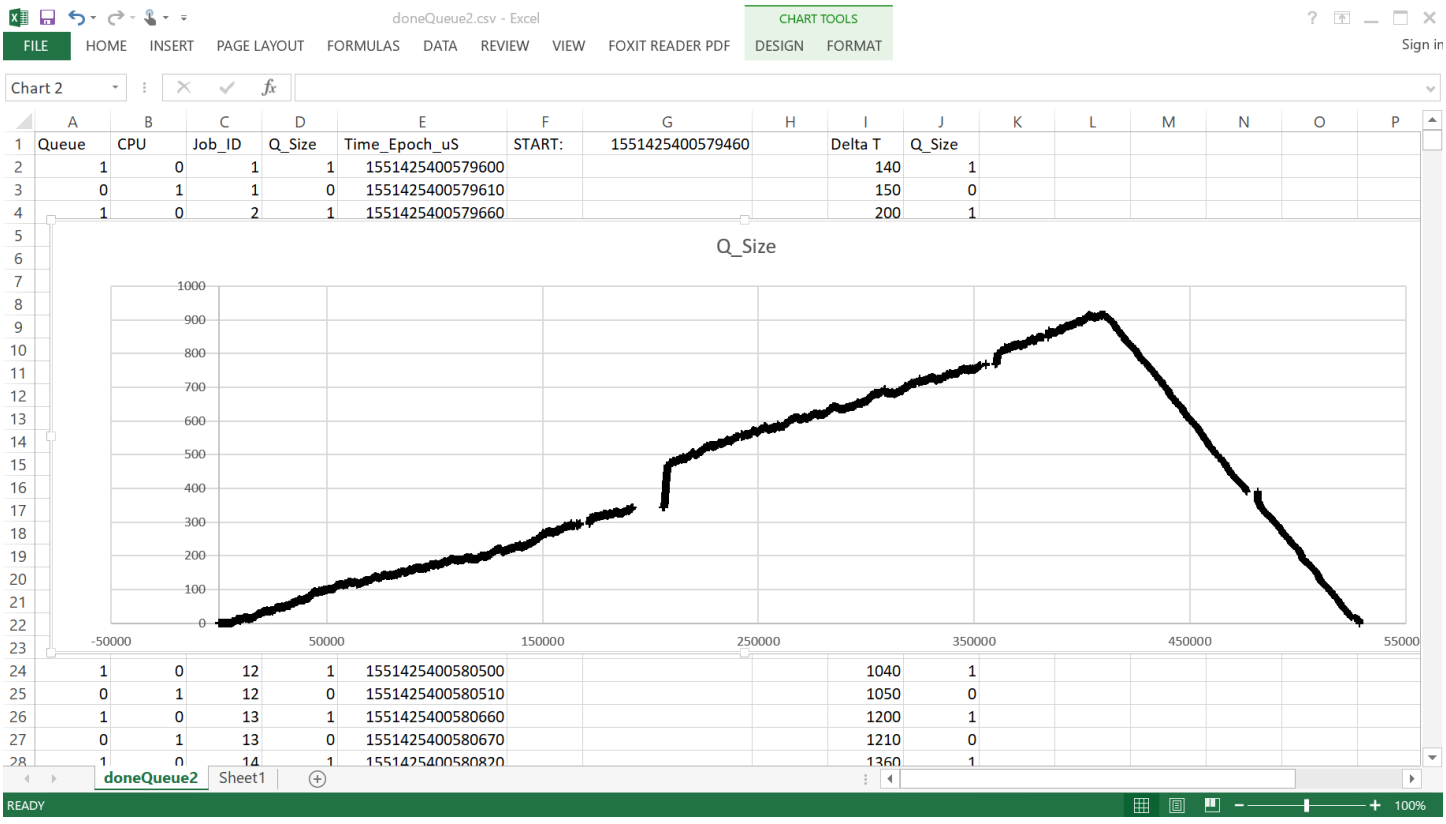




4,000 job run completed successfully.

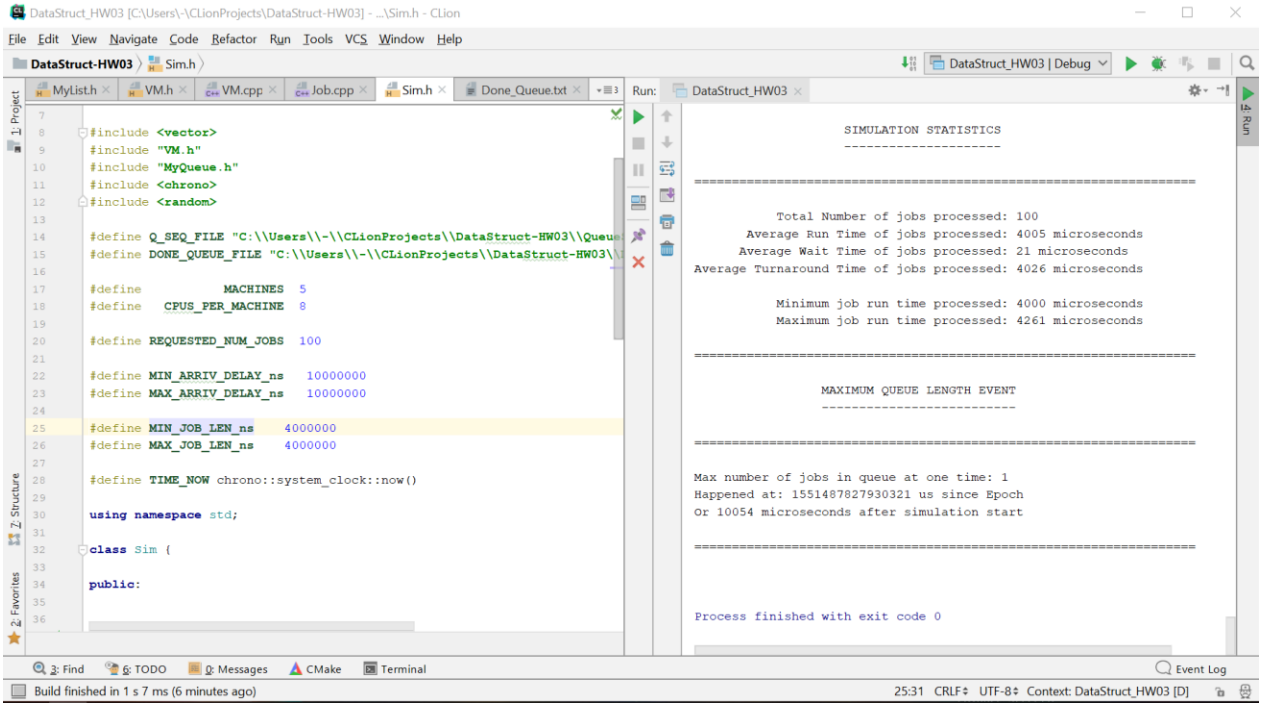


Another graph of output file values showing a break, indicating the computer was doing something else during simulation

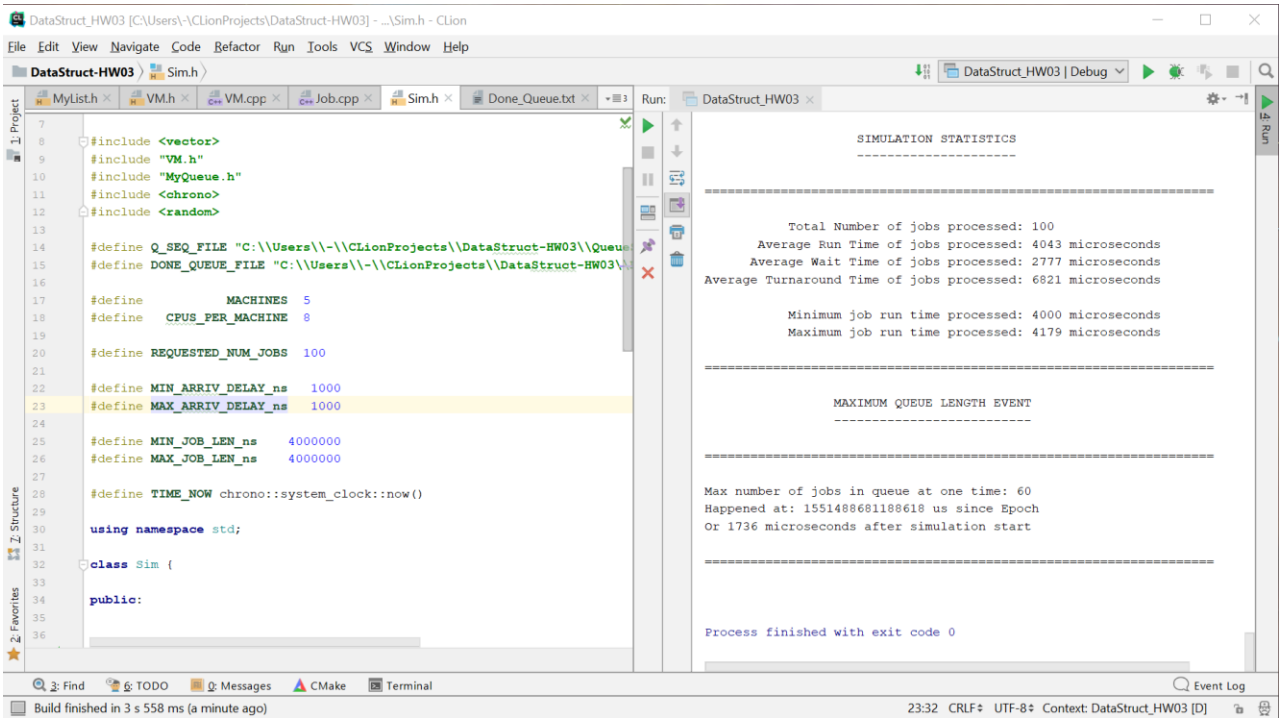


For Testing: set high and low limits of random values to the same value in order to test cumulative and averaging functions for reasonable values.

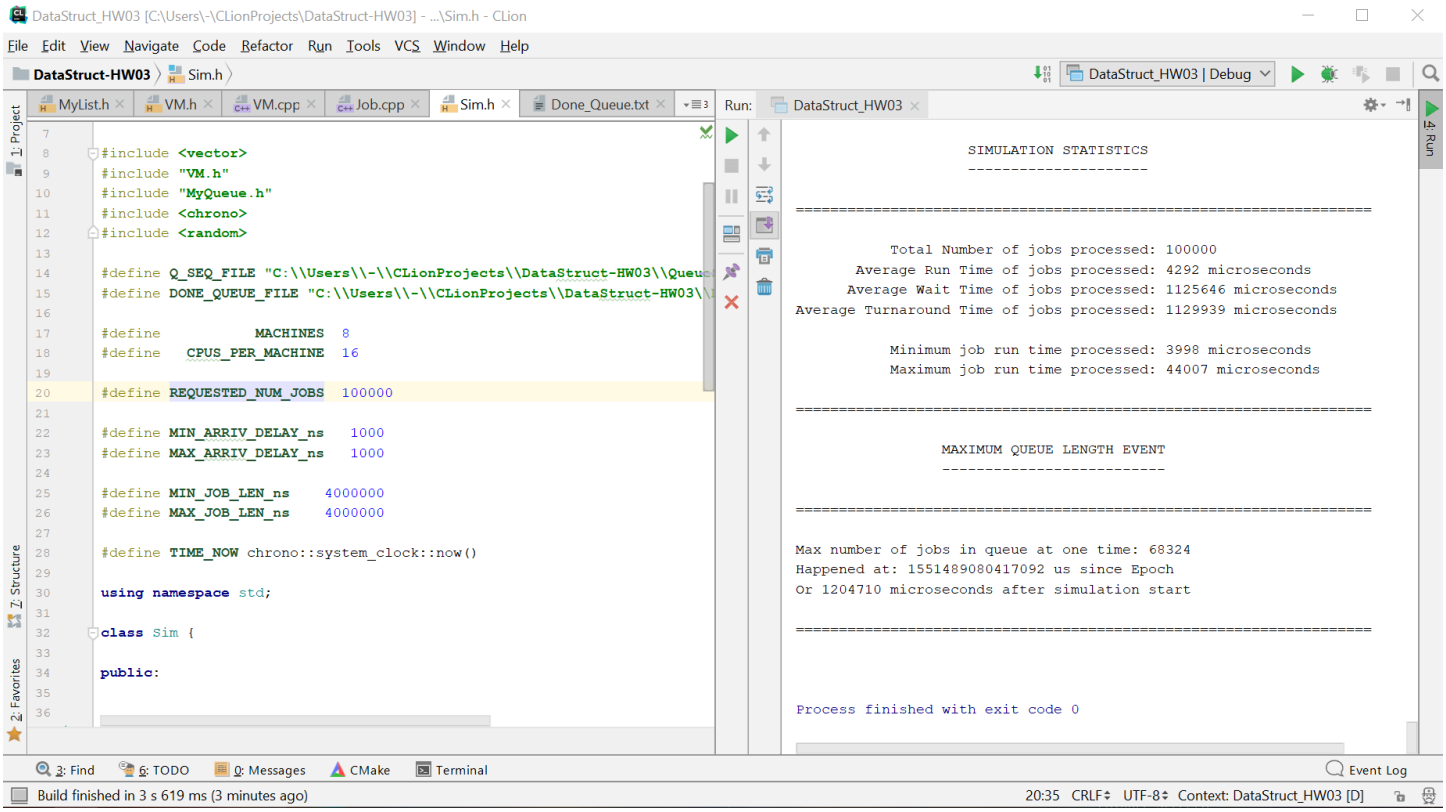
Set delay values high to eliminate any waiting in queue and set requested job length to 4 milliseconds. Maximum queue length correctly reported as 1, Average runtime correct at 4.005 milliseconds, Average wait time fairly close to 0 (.021 milliseconds), Average Turnaround time correct at 4.026 milliseconds, Minimum job length correct, Maximum job length correct and indicates other processes are happening on computer.



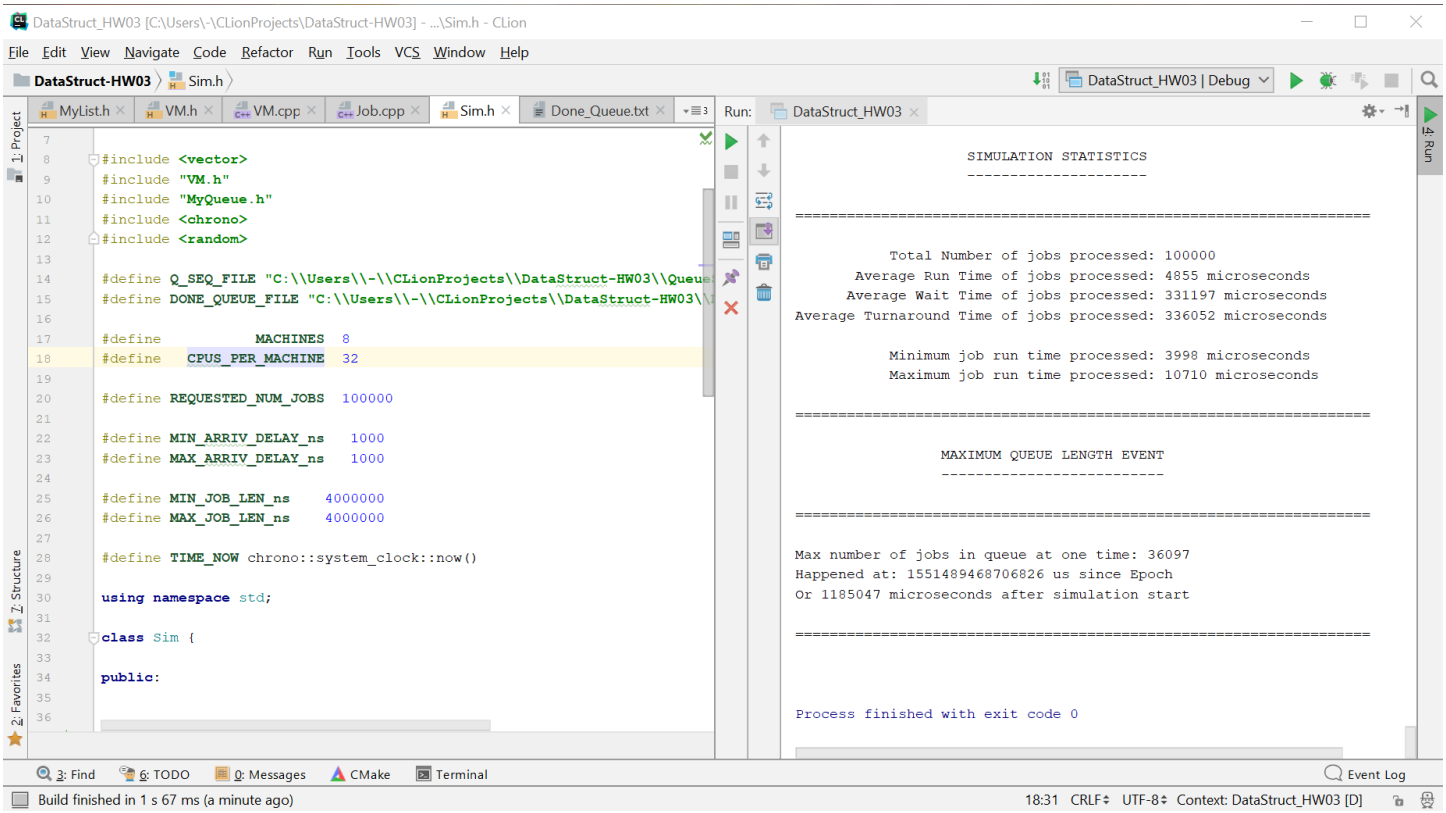
Set delay values to 1 microseconds to ensure high queue congestion (built up to 60), left job length at 4 milliseconds. Maximum queue length looks reasonable, Average runtime correct at 4.043 milliseconds, Average wait time increased to 2.8 milliseconds, Average Turnaround time value within expected values, Minimum job length correct, Maximum job length within expected values.



Requested 100,000 jobs processed on 8 virtual machines with 16 CPUs each. Queue congestion caused a maximum of 68,324 jobs waiting in queue.



Doubled the CPUs to 8 machines of 32 CPUs. Reduced wait queue max size to 36,097 jobs.



Graph of queue size (building up to 36,097) vs time using file output of QueueSequence.csv .