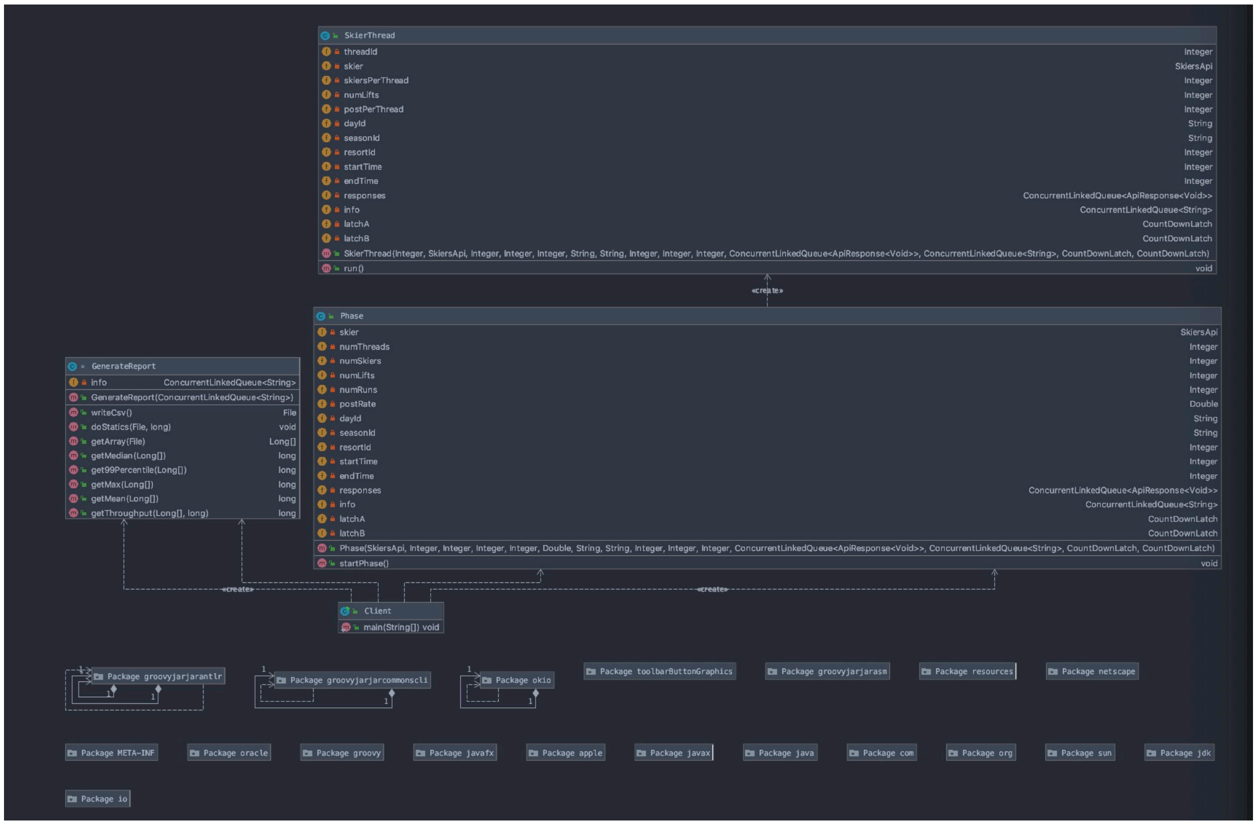
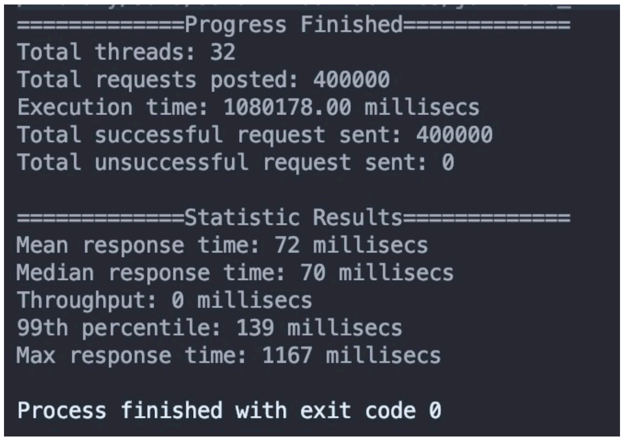
URL of GitHub link:
https://github.ccs.neu.edu/yihtian/CS6650/tree/master/Assignment1

Design
For the client side, 3 classes are designed (with one class to generate csv file and reports in part2, 4 classes in total submission).
First class is SkierThread. This class represents a single thread of the program. It takes in parameter of all rides' information and initializes and creates a single thread of the ski resort, and the run method is override and allow a single thread of client to send required number of posts to server. 2 count down latches are created and counted down in this part, latch 1 is designed to count a percentage of a single phase and determine when next phase is able to run, latch 2 is designed to count for a whole phase and block further operations until all threads in a single phase are end.
Nest class is Phase, it represents a single phase that a client will run. It also takes in parameter of all ride's information, and initialize a phase or client, which will create required number of threads and start running the threads.
Last class of client is Client. It contains only a main function, which get inputs of requirements, and start 3 phases of task. As required, when 10% threads of phase 1 are end phase 2 starts, and when 10% percent of threads end phase 3 will start. After all threads of total 3 phases are end, the report and statistics will begin to conduct.
The extra class, GenerateReport, is a separate class which provides methods that allows a client to write out information into csv files and do statistics.
Relations between classes are described as a UML as below:



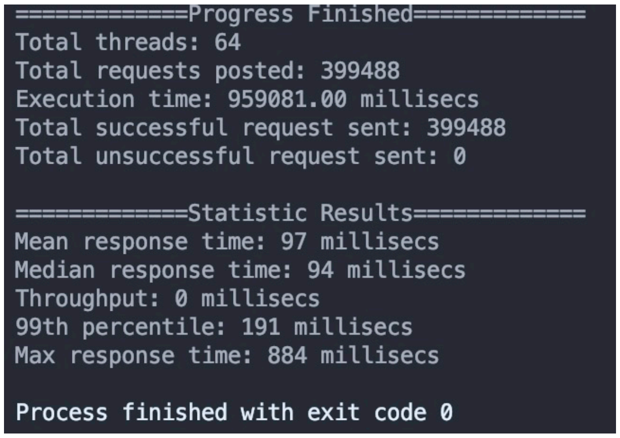
Outputs for each Runs (Both for client part 1 and part 2)

For 32 threads:



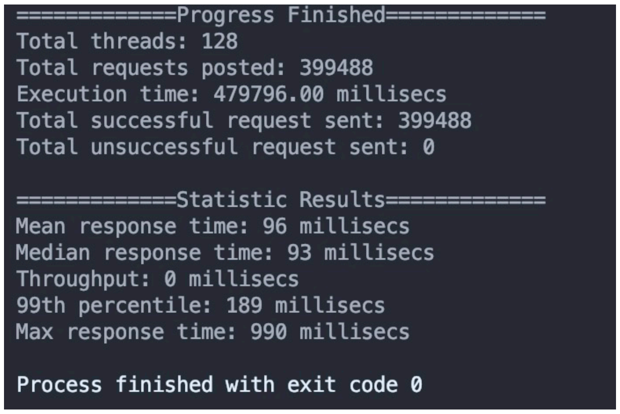
```
=============Progress Finished=============
Total threads: 32
Total requests posted: 400000
Execution time: 1080178.00 millisecs
Total successful request sent: 400000
Total unsuccessful request sent: 0

=============Statistic Results=============
Mean response time: 72 millisecs
Median response time: 70 millisecs
Throughput: 0 millisecs
99th percentile: 139 millisecs
Max response time: 1167 millisecs

Process finished with exit code 0
```

For 64 threads:



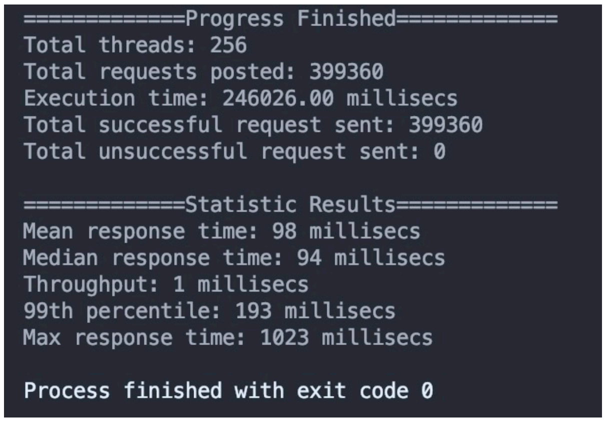
```
=============Progress Finished=============
Total threads: 64
Total requests posted: 399488
Execution time: 959081.00 millisecs
Total successful request sent: 399488
Total unsuccessful request sent: 0

=============Statistic Results=============
Mean response time: 97 millisecs
Median response time: 94 millisecs
Throughput: 0 millisecs
99th percentile: 191 millisecs
Max response time: 884 millisecs

Process finished with exit code 0
```

For 128 threads:



```
=============Progress Finished=============
Total threads: 128
Total requests posted: 399488
Execution time: 479796.00 millisecs
Total successful request sent: 399488
Total unsuccessful request sent: 0

=============Statistic Results=============
Mean response time: 96 millisecs
Median response time: 93 millisecs
Throughput: 0 millisecs
99th percentile: 189 millisecs
Max response time: 990 millisecs

Process finished with exit code 0
```

For 256 threads:

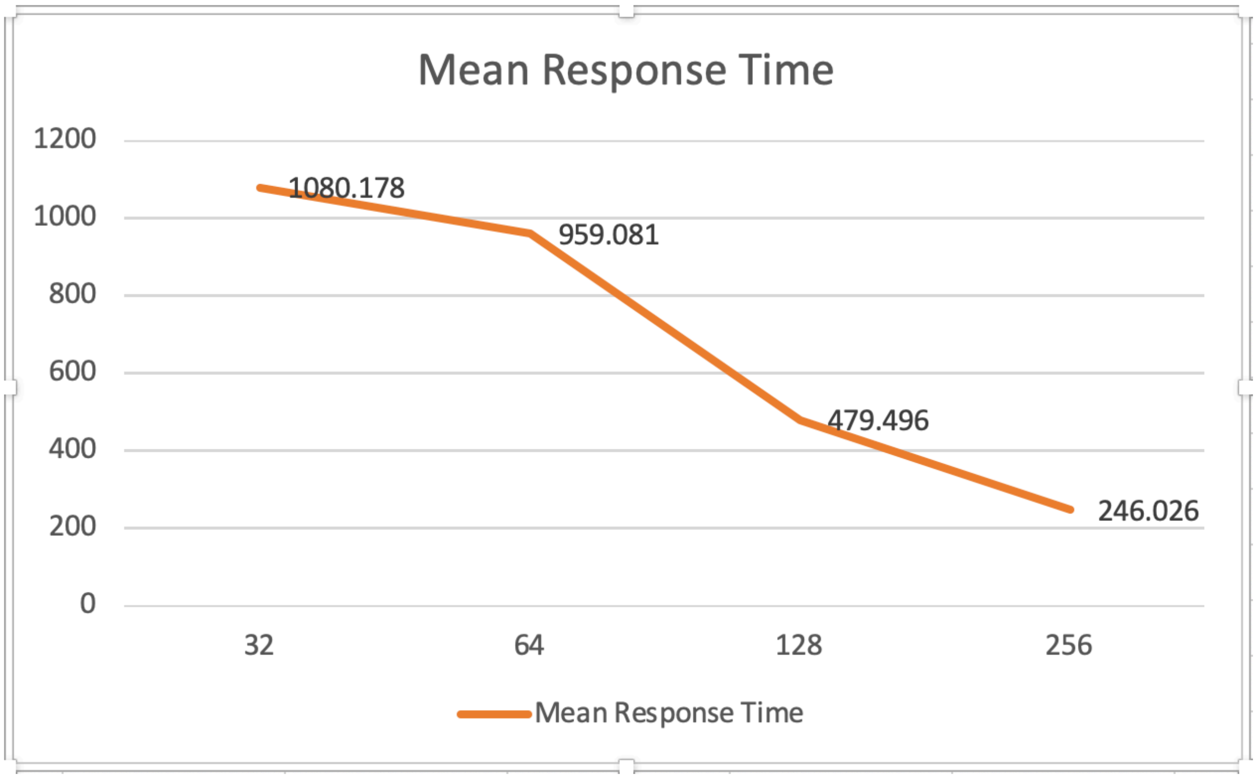


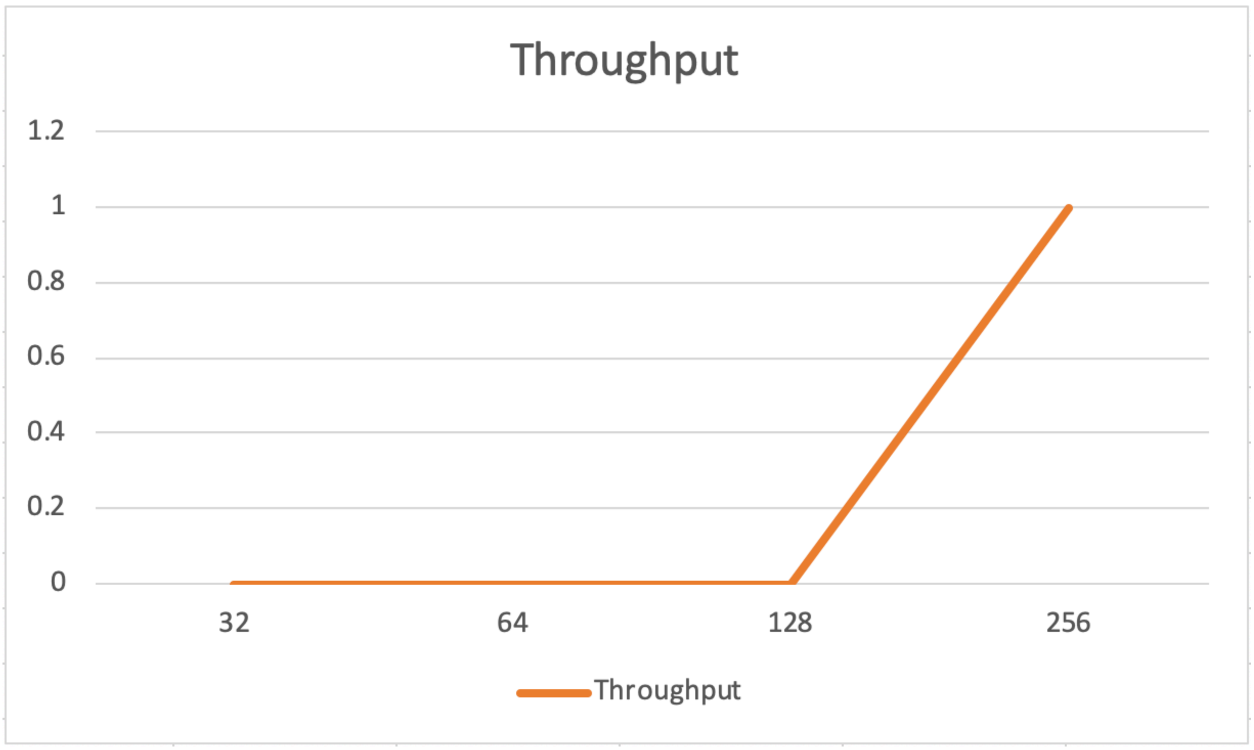
```
=============Progress Finished=============
Total threads: 256
Total requests posted: 399360
Execution time: 246026.00 millisecs
Total successful request sent: 399360
Total unsuccessful request sent: 0

=============Statistic Results=============
Mean response time: 98 millisecs
Median response time: 94 millisecs
Throughput: 1 millisecs
99th percentile: 193 millisecs
Max response time: 1023 millisecs

Process finished with exit code 0
```

Wall time by threads:

32 Threads: 1080.178 s (18 min)
64 Threads: 959.081 s (16 min)
128 Threads: 479.496 s (8 min)
256 Threads: 246.026 s (4 min)

Plot:


Mean Response Time
1200
1000
800
600
400
200
0
1080.178
959.081
479.496
246.026
32
64
128
256
Mean Response Time


Throughput
1.2
1
0.8
0.6
0.4
0.2
0
32
64
128
256
Throughput