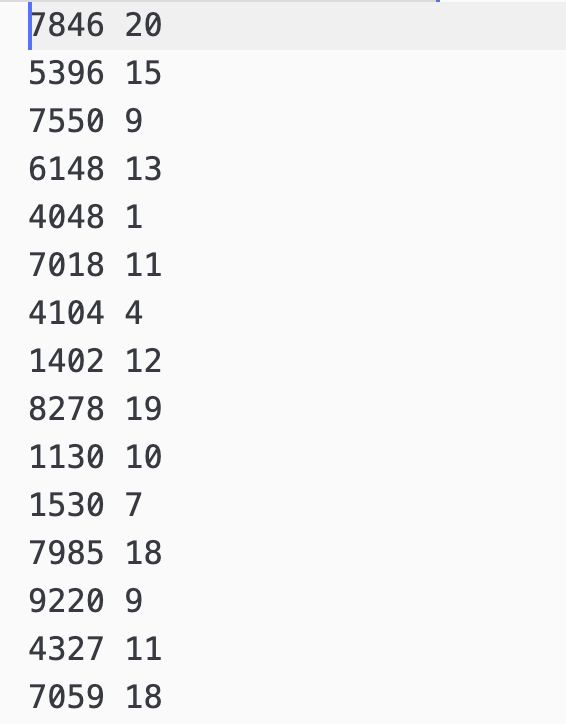
**CSCI 3400 Test 2 Assignment**

In this assignment, you will write a program that simulates a simple version of the process scheduling system in the computer processor. Your program will read a set of process information from an input file (processList.txt) and schedule their execution order using a Queue data structure. Every process in the file has an ID and a required time of execution (in ms). A sample input file is shown below:



As an example of above file, the process with ID=5396 needs 15 ms execution time. Similarly, the process with ID=7550 needs 9 ms execution time, etc. When a process in the queue is executed, its remaining execution time is reduced by 1ms. So, every process is expected to be executed multiple times until their remaining execution time is zero.

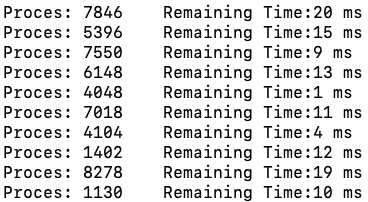
Initially, you will create a class called “Process”. This class will have two attributes and several member functions. Please see details of the class below:

|  |  |  |
| --- | --- | --- |
| **Process** | | |
| ***Type*** | ***Attribute*** |  |
| int | ID | 4 digit process ID |
| int | execTime | Remaining execution time of the process |
| ***Return Type*** | ***Function*** | ***Description*** |
| (constructor) | Process(int newID, int newTime) | Initialize the object |
| void | execute() | Simulates the execution of the process. Subtracts 1 from execTime attribute. |
| int | getID() | Returns process ID |
| int | getTime() | Returns execTime |
| void | Print() | Prints process ID and execTime on the screen |

Once your class is ready, you will create a queue of “Process”. You will use C++ STL Queue library for this queue object. You do not have to design a new queue class. See the reference link below:

<https://www.geeksforgeeks.org/queue-cpp-stl/>

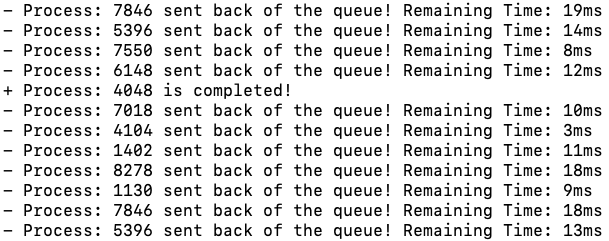
Please note that function names of the STL queue class are not matching with the names of queue we implemented in class. Please use the STL function names accordingly. After creating the process queue, you will load all processes in file to the queue. First, you will print all processes and their execution times. For example:



Then the execution starts. In a while loop (while the queue is not empty), you will dequeue() (or pop() in the STL version) a process from the queue and execute it using the process execute() function. This execution is a very simple simulation, where only the execution time of the process is reduced by 1ms. If the remaining execution time is still greater than zero, then the process is sent back of the queue using the enqueue() (or push() in the STL version) function. If the remaining time is zero, “the process complete” message is printed on the screen. This way, in every iteration of the while loop, the process at the beginning of the queue is removed, executed, and added back (if not completed). See the following pseudo-code for the while loop:

|  |
| --- |
| while(process queue is not empty){  Take the process at the head of the queue using front() and pop()  Execute the process  if(the process time equals zero)  Print “Process with ID=X is completed”  Else  Print “Process with ID=X needs Y more ms of execution.”  Send the Process to the back of the queue using push()  Print “Process ID=X is sent back of the queue”  } |

A sample screen output of a few execution steps is given below.



…

This while loop will run until all processes are completely executed. Then your program will end. I am expecting you to submit a single cpp file including everything. Please let me know if you have any questions.