In order to modify the original map file into a multimap, I changed the value v into a queue of Vs. I chose a queue because we were suppose to remove from the front and insert into the back, a situation perfect for a queue. In the get function, I changed the return from value to value.front(). For the Insert function, I removed the error you would usually get if a key was already present and instead had it push a value in the queue. For the remove function, under the point where the key is found, I added a portion that looks to see if the number of values is greater than one. If it is, it will pop a value. If not, it will run as normal. I also added a condition at the end that checks if the key is still present in order to determine if it fixes the tree. In order to test the new multimap header, I adapted the two tests in the original map test and added tests to check the multimap's ability to map multiple values to one key, how the max and min function work even when removing values, ensuring they would update. In addition, I added a test to ensure the size function would work both when adding more keys and values to said keys. Finally, I added a test that ensured that only the first value for each key was returned.
In order to implement the cfs, I began by creating a vector of pointers to my custom class, Task. I chose to use a vector for two major reasons. One reason was the sort function, which allowed me to sort the pointers. In order to do this, I used the std::sort function from algorithm, writing a custom function that took into account their start time and character identifier. The second reason was because of the vector's ability to delete from anywhere inside of it. Because I didn't know which processes would finish first, the versatility offered by the vector was extremely appealing. After creating a sorted vector, I then created a multimap that would act as the timeline for the process. In order to begin the loop, I chose to go off of the condition "while array is not empty". This would ensure that the loop would continue until every process was actually done. From there, the while loop iterates over the instructions mentioned in the homework, deleting and removing the pointer from the vector whenever the task it represents is finished, and printing out each step it takes.