Joseph Lewis

Jaycob Callies

PA 5 Design document

**Part 1:**









-Blue Arrows are Housekeeping Functions

-Black Arrows are Event-driven functions

-Green rectangle are data containers

**Part 2 Time Complexities:**

InsertJob: O(log(n))

Possible to be inserted all the way to the lowest priority.

FindShortest: O(1)

Checks availability, removes processors from pool, inserts to active, removes from waiting.

DeleteShortest: O(1)

Simply removes the value at top.

CheckAvailability: O(1)

Only two comparisons are made, independent of size

RunJob(): O(1)

The singular decrement of a job timer

DecrementTimer: O(n);

The decrement of all job timers requires traversal.

ReleaseProcs: O(n)

Traverses in search of completed jobs to delete.

**Part 3:**

The use of a priority queue itself along with the way it is implemented in out code has its shortcomings, in general as well as in this problem. In general the downside of a priority queue is that is does not allow for traversal or modification of data other than viewing top, removing top, and inserting. In this case it means that the priority queue must be made so it orders opposite, with the lowest on top. A downside in this problem is that the lowest time job first is not necessarily the quickest way to solve all jobs, if the wait queue was ordered to have the lowest processors required on top instead of lowest time required it would be able to complete the entirety of the wait queue faster as the maximum amount of processors would remain active at all times.

Note: Main() is defined in TestDriver.cpp as well as Main.cpp, only include the appropriate file when compiling.