Describe each of the stub functions you modified and how you tested the scheduler and any difficulties you had in implementing the project as well as any functionality that does not completely work or behaves unexpectedly.

enqueue\_task\_other\_rr() takes the given task and first examines whether the task given is woken up and runnable. Next the task is added to the end of the run queue to await its turn for execution. Finally the run queue is informed to increment the number of currently runnable tasks it is responsible for.

dequeue\_task\_other\_rr() deletes the given task from the queue and decrements the number of runnable tasks in the run queue.

yield\_task\_other\_rr() finds the head and currently running task in the given run queue. From this the function then finds the list entry representing the current task. The CPU is then given to the next task in the queue and the old current task is requeued into the end of the line.

pick\_next\_task\_other\_rr()

task\_tick\_other\_rr()

One difficulty we encountered while trying to test our code was that while our kernel compiled and installed, it would crash upon booting without producing any error messages to explain what happened. We reinstalled a fresh version of our kernel and migrated our changes over one stub function at a time to determine the cause of the crash.