RTX P1

Things we have learned and mistakes we have made

- **Always initialize your variables, especially pointers = NULL**. When variables are not initialized, this mistake is hard to find. On the simulator, uninitialized variables work fine because the simulator either catches the exception for you or sets the variable to NULL initially. However, the board does not do this for you and will give you garbage values that will be used and cause more issues.
- In this project, the scheduler handles the selection of which process should be executed next. In our initial implementation, when calling the scheduler from release_processor we would take a process from the ready queue without first pushing the current process onto the queue. We changed it so that we **pushed our process into our ready queue before calling the scheduler** (which could potentially select the current process to run again).
- **We realized that we cannot allocate memory easily, because the OS must be responsible for managing that for other programs**. We ran into this problem when trying to allocate memory for our priority queue and heap. It was a new experience to allocate memory at such a low level where we actually had to move a pointer rather than calling a function that allocates for us.
- **Realizing our OS code's stack pointer is the same stack pointer as the user process's.** At the very beginning we did not understand this and it took some time to figure out.
- **Different configurations are required for board and the simulation**
- We initially structured our heap as a linked list of pointers. However, we found out later that this structure made it difficult to keep track of memory locations. It was difficult because things would get popped and pushed on in different order making it hard to check whether a block was valid. Hence, we used a different method which utilizes a buffer to store the states of these blocks of memory and uses the memory as an array rather than a linked list.
- **Unit tests are extremely useful and helpful**, especially as a sanity check for simple functions and data structures.