For the FCFS portion, since FCFS prioritizes processes based on their arrival time. In the code I made sure processes are initialized and added to a ready queue in order of arrival time. Then made sure it maintains a running process, which is the one currently being executed. At each cycle, it checks if there's a process that's currently being executed. If there isn't a current process, it selects the next process from the queue to execute. The tie-breaking rule for FCFS is whenever processes have the same arrival time, they are executed in the order they appear in the queue.

For the Round Robin portion, I had processes added to a queue based on their arrival time. It maintains a running process and uses a fixed time quantum for each process. Then during each cycle, it checks if the current process's quantum has expired. If it has, the process is moved to the back of the queue, allowing the next process to execute. If the quantum hasn't expired, the process continues execution. The tie-breaking rule for Round Robin is implemented by selecting the process in the front of the "ready" queue is chosen, when selecting the next process to run after a quantum expires.

For the Shortest Job First portion, I made sure that the processes are added to a "ready" queue based on their arrival time. The code maintains a "running" process, and at each cycle, it selects the process with the shortest remaining burst time from the "ready" queue. The tie-breaking rule for Shortest Job First is when two processes have the same remaining burst time the process with the earliest arrival time is selected.