Output format:

For each job enqueued/dequeued, print out the following:

```
Producer <process-id> added <job_size> to buffer

Consumer <consumer-id> dequeue <process-id, job_size> from buffer
```

Also print out the total execution time at the end of your code.

Report format:

1) First give an outline of the working portions of your code. For each part not functioning list them out.
2) Second: show the following: (a) logic used to identify the terminating condition, (b) how were the semaphores and other book-keeping variables shared between processes and threads, (c) what was done in the signal handler for graceful termination and (d) how did the FCFS and SJF implementations differ in terms of your usage of the buffer_index variable from the producer-consumer code of Assignment-2.
3) Third: show two sample runs of your code one for FCFS and other one for SJF. Just copy-paste the code output from the server.
4) Finally, show the execution time plots for FCFS and SJF for different values of number of producer processes and number of consumer threads. Plots will be good to show here or you can simply use a table format; vary both #producers and #consumers.
5) Two options:
    a. 3-D graph: you will just have a single 3-D graph for FCFS, and another 3-D graph for SJF.
    b. 2-D graphs: 5 different graphs are needed for #producers = 2, 4, 6, 8, 10 respectively. In each graph, plot execution time VS number of consumers (varied as 2, 4, 6, 8, 10) for FCFS and for SJF. So each 2-D graph will have two lines, one for FCFS and other one for SJF.

Turn-in:

Submit two versions of your code: one for FCFS, other one for SJF. Keep the output format (from above) the same for both implementations. Additionally, submit the report. All files must be uploaded through BlackBoard.

Bonus points:
You can get 20 bonus points if you implement the extension where each queue element is controlled by a separate binary semaphore. You need 30 additional binary semaphores to do this. Show the execution time improvements for this case.