Eric Smith
April 19th, 2017

CSE7343 – Semester Project
Phase 2

Introduction

My semester project was written in C++. I chose this language because it's the one I love most! I also have past experience using C to create the necessary data structures.

Compilation Instructions

This project was compiled and tested on both Linux and MacOS machines.

To run the program, simply compile all .CPP files using the C++11 standard compiler. Then, call the resulting executable.

```
Erics-MacBook-Pro:CSE7343 - Semester Project ericsmith$ g++ -std=c++11 *.cpp
Erics-MacBook-Pro:CSE7343 - Semester Project ericsmith$ ./a.out
```

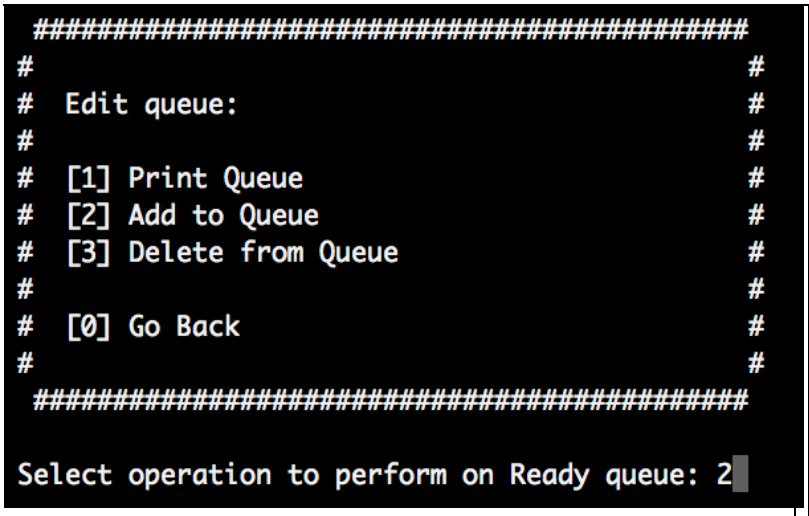
The Queues

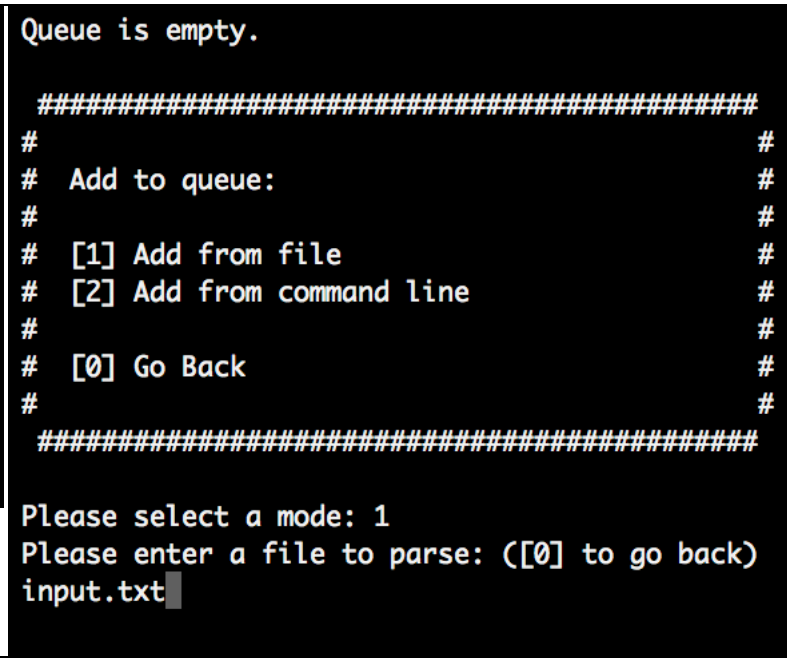
The application has two queues: the waiting and ready queues. The user has the ability to add to, delete from, and print both queues. However, the waiting queue only exists as a proof of concept. Those processes that are added to the *ready* queue will be the ones that are executed under "[3] Execute Processes" and "[4] Calculate Memory Usage".

To add to the ready queue, enter "1" when prompted at the main menu. You will be shown the three options for interacting with the ready queue: print, add, and delete.

```
 ##########################################
#  Welcome to EricOS                       #
#  Version 2.0                             #
#                                          #
#  Main Menu:                              #
#                                          #
#  [1] Edit Ready Queue                    #
#  [2] Edit Waiting Queue                  #
#  [3] Execute Processes                   #
#  [4] Calculate Memory Usage              #
#                                          #
#  [0] Exit                                #
#                                          #
 ##########################################

Please select a mode: 1
```

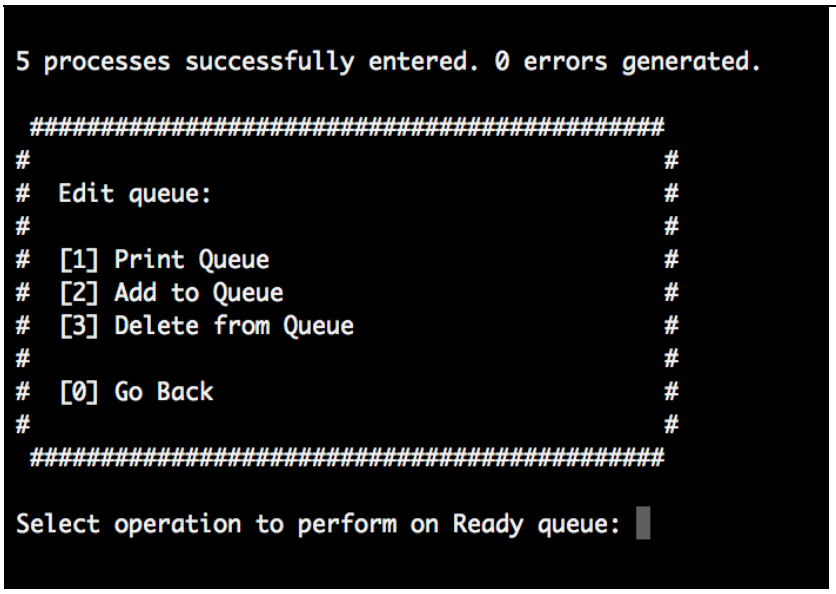


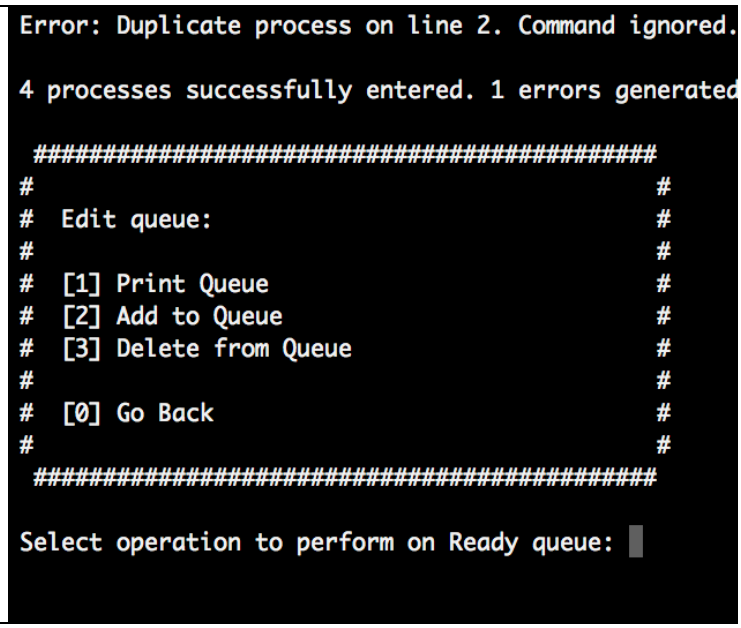


When you enter “2”, if any processes are in the ready queue, they will be printed out. You’ll then have the choice of adding to the queue from a file or the command line. Above is an example of adding from a file.

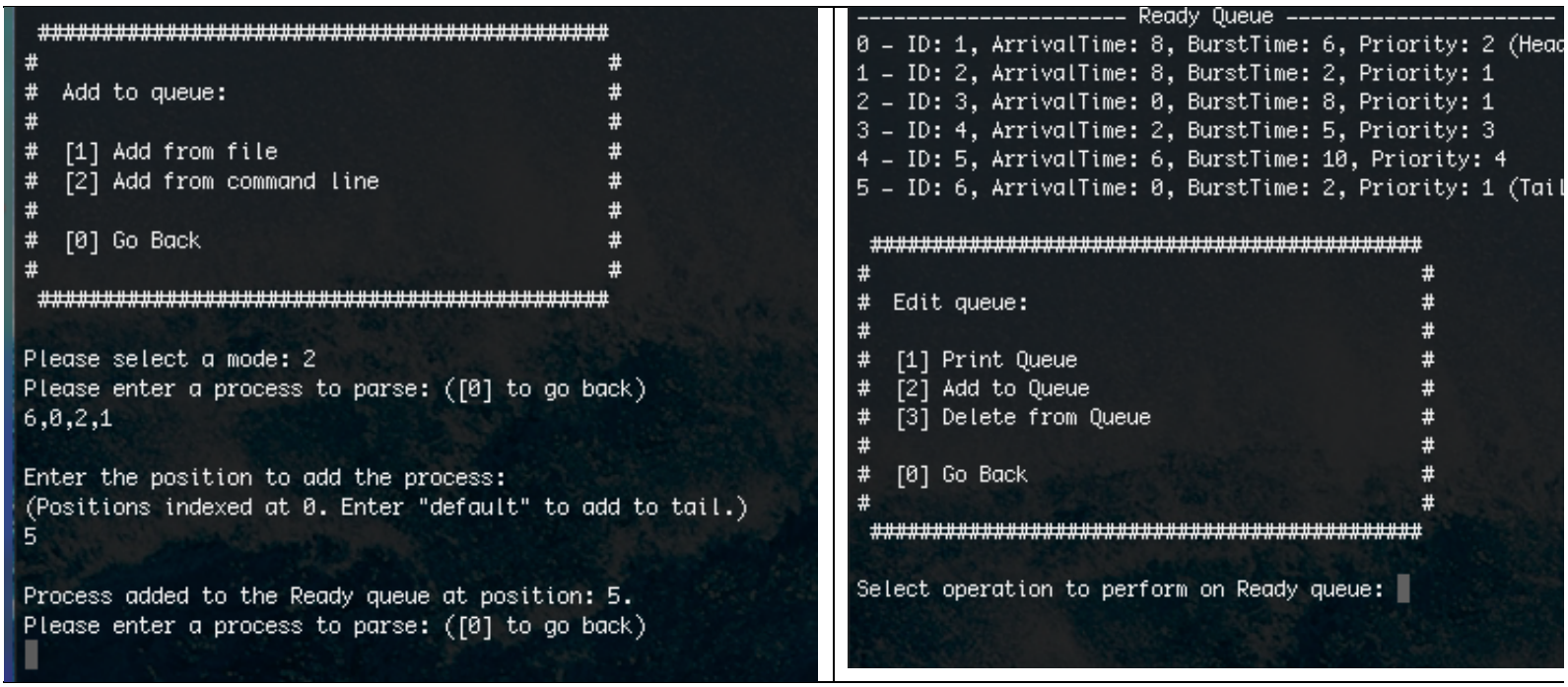
Please enter processes in the following format:
Process ID (1-99999), Arrival Time (0-9999), Burst Time (0-9999), Priority (1-4), Memory (0-9999)

If there are any errors in the file passed to the program, they will be displayed above the menu. All other processes’ whose lines were valid will be loaded into the ready queue.

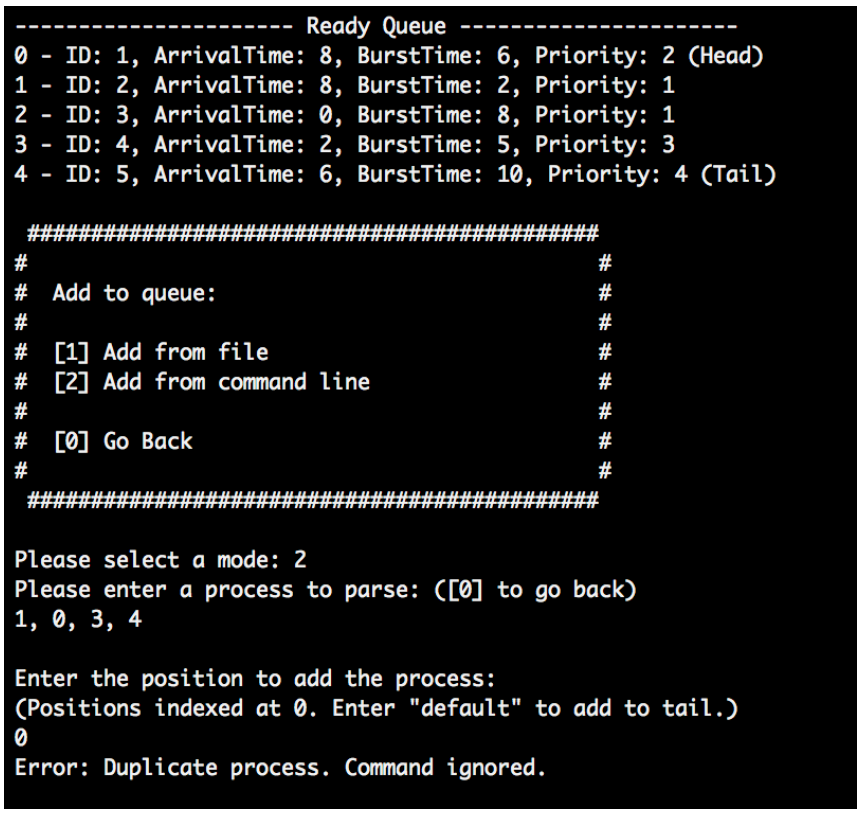




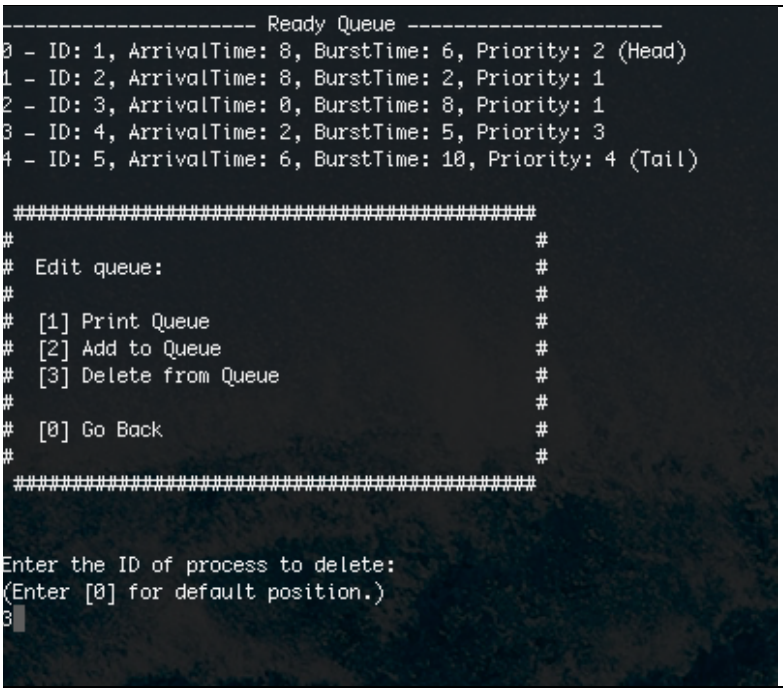
Processes can also be added from the command line. From the edit queue menu, press “2” again to add to the queue, then select “2” again to add from the command line. You’ll then have the option to add the process at an index of the queue (starting at zero) or add to the tail of the queue by entering “default”.

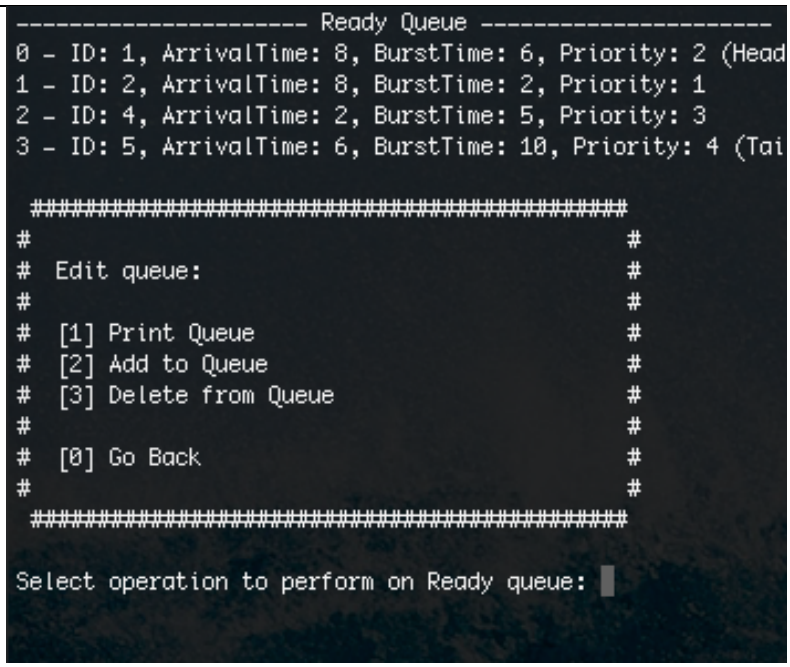


The same error checking rules apply to the command that apply to entering a file, any logical or syntactical errors will prevent the process from being added to the queue.



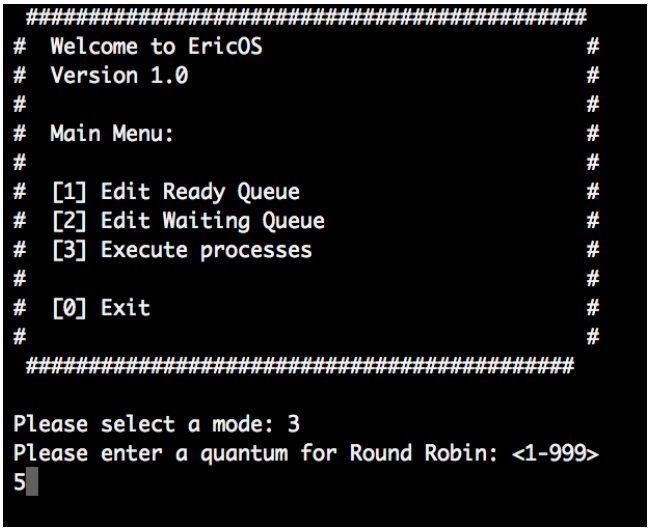
To delete from a queue, select “3” from the edit queue menu. You can then enter the ID of the process to delete. (Process 3 was deleted in the example below)





Scheduling Algorithms

Now that the ready queue has some processes, let’s run the scheduling algorithms on the ready queue. Choose “3” at the main menu to run the processes, then enter a desired time quantum for round robin.



You'll then be shown the results of all the scheduling algorithms at the same time. Please note that SJF, FCFS, and Priority scheduling all happen NON-preemptively. In the case of first come first serve, processes are chosen by arrival time. If two arrival times are the same, priority is used as a tie breaker. For round robin, the next process to have arrived is selected to run. The column "Wait Time" shows the wait time with that individual process burst. The column "Total Wait Time" shows the wait time associated with the process as a whole.

```
------------------ Shortest Job First -------------------
Process         Start Time       End Time        Wait Time
1 - P3           0               8               0
2 - P2           8               10              0
3 - P4           10              15              8
4 - P1           15              21              7
5 - P5           21              31              15

Average waiting time: 6

------------------ First Come First Serve ------------------
Process         Start Time       End Time        Wait Time
1 - P3           0               8               0
2 - P4           8               13              6
3 - P5           13              23              7
4 - P2           23              25              15
5 - P1           25              31              17

Average waiting time: 9

----------------------- Priority ------------------------
Process         Start Time       End Time        Wait Time
1 - P3           0               8               0
2 - P2           8               10              0
3 - P1           10              16              2
4 - P4           16              21              14
5 - P5           21              31              15

Average waiting time: 6.2

--------------------------------- Round Robin ---------------------------------
Process         Start Time       End Time        Wait Time       Total Wait Time
1 - P3           0               5               0               17
2 - P4           5               10              5               3
3 - P5           10              15              10              14
4 - P1           15              20              15              17
5 - P2           20              22              20              12
6 - P3           22              25              17              17
7 - P5           25              30              10              14
8 - P1           30              31              10              17


Average wait time: 12.6

Enter [0] to go back.
```

Calculating Memory Usage

To calculate memory usage, select option 4 at the Main Menu. (Processes must first be entered into the ready queue.)

You'll be prompted to enter the total memory size and the number of available memory blocks, which will be used for some of the statistics. For each memory block, you'll be asked to manually enter the size of the block.

```
 ##########################################
#  Welcome to EricOS                       #
#  Version 2.0                             #
#                                          #
#  Main Menu:                              #
#                                          #
#  [1] Edit Ready Queue                    #
#  [2] Edit Waiting Queue                  #
#  [3] Execute Processes                   #
#  [4] Calculate Memory Usage              #
#                                          #
#  [0] Exit                                #
#                                          #
 ##########################################

Please select a mode: 4
Please enter the total memory size: <1-99999>
4000
Please enter the number of available memory blocks: <1-999>
6
Please enter the size of memory block Number 1 : <1-9999>
300
Please enter the size of memory block Number 2 : <1-9999>
600
Please enter the size of memory block Number 3 : <1-9999>
350
Please enter the size of memory block Number 4 : <1-9999>
200
Please enter the size of memory block Number 5 : <1-9999>
750
Please enter the size of memory block Number 6 : <1-9999>
125
```

You'll then be showed memory usage data and statistics for each algorithm:

```
---------------------------------------------------- First Fit ----------------------------------------------------

t=0:     Adding process P1 to memory block 1.          (Memory block 1 now has 185 units of available space.)
t=0:     Adding process P2 to memory block 2.          (Memory block 2 now has 100 units of available space.)
t=0:     Adding process P3 to memory block 5.          (Memory block 5 now has 392 units of available space.)
t=0:     Adding process P4 to memory block 3.          (Memory block 3 now has 150 units of available space.)
t=0:     Adding process P5 to memory block 5.          (Memory block 5 now has 67 units of available space.)
t=10:    Process P1 has completed.                     (Memory block 1 now has 300 units of available space.)
t=10:    Process P2 has completed.                     (Memory block 2 now has 600 units of available space.)
t=10:    Process P4 has completed.                     (Memory block 3 now has 350 units of available space.)
t=10:    Process P3 has completed.                     (Memory block 5 now has 425 units of available space.)
t=10:    Process P5 has completed.                     (Memory block 5 now has 750 units of available space.)
------------------------------------------------------------------------------------------------------------------
5 processes were loaded into and out of memory with 0 fragmentations.
Blocking probability: 0%
Total wait time of fragmented processes: 0
Maximum memory utilization: 79% occurs at t=0
------------------------------------------------------------------------------------------------------------------


---------------------------------------------------- Best Fit ----------------------------------------------------

t=0:     Adding process P1 to memory block 6.          (Memory block 6 now has 10 units of available space.)
t=0:     Adding process P2 to memory block 2.          (Memory block 2 now has 100 units of available space.)
t=0:     Adding process P3 to memory block 5.          (Memory block 5 now has 392 units of available space.)
t=0:     Adding process P4 to memory block 4.          (Memory block 4 now has 0 units of available space.)
t=0:     Adding process P5 to memory block 3.          (Memory block 3 now has 25 units of available space.)
t=10:    Process P2 has completed.                     (Memory block 2 now has 600 units of available space.)
t=10:    Process P5 has completed.                     (Memory block 3 now has 350 units of available space.)
t=10:    Process P4 has completed.                     (Memory block 4 now has 200 units of available space.)
t=10:    Process P3 has completed.                     (Memory block 5 now has 750 units of available space.)
t=10:    Process P1 has completed.                     (Memory block 6 now has 125 units of available space.)
------------------------------------------------------------------------------------------------------------------
5 processes were loaded into and out of memory with 0 fragmentations.
Blocking probability: 0%
Total wait time of fragmented processes: 0
Maximum memory utilization: 79% occurs at t=0
------------------------------------------------------------------------------------------------------------------


---------------------------------------------------- Worst Fit ----------------------------------------------------

t=0:     Adding process P1 to memory block 5.          (Memory block 5 now has 635 units of available space.)
t=0:     Adding process P2 to memory block 5.          (Memory block 5 now has 135 units of available space.)
t=0:     Adding process P3 to memory block 2.          (Memory block 2 now has 242 units of available space.)
t=0:     Adding process P4 to memory block 3.          (Memory block 3 now has 150 units of available space.)
t=0:     Fragmentation. Not enough room to add P5.
t=1:     Fragmentation. Not enough room to add P5.
t=2:     Fragmentation. Not enough room to add P5.
t=3:     Fragmentation. Not enough room to add P5.
t=4:     Fragmentation. Not enough room to add P5.
t=5:     Fragmentation. Not enough room to add P5.
t=6:     Fragmentation. Not enough room to add P5.
t=7:     Fragmentation. Not enough room to add P5.
t=8:     Fragmentation. Not enough room to add P5.
t=9:     Fragmentation. Not enough room to add P5.
t=10:    Fragmentation. Not enough room to add P5.
t=10:    Process P3 has completed.                     (Memory block 2 now has 600 units of available space.)
t=10:    Process P4 has completed.                     (Memory block 3 now has 350 units of available space.)
t=10:    Process P1 has completed.                     (Memory block 5 now has 250 units of available space.)
t=10:    Process P2 has completed.                     (Memory block 5 now has 750 units of available space.)
t=11:    Adding process P5 to memory block 5.          (Memory block 5 now has 425 units of available space.)
t=21:    Process P5 has completed.                     (Memory block 5 now has 750 units of available space.)
------------------------------------------------------------------------------------------------------------------
5 processes were loaded into and out of memory with 11 fragmentations.
Blocking probability: 0%
Total wait time of fragmented processes: 11
Maximum memory utilization: 71% occurs at t=0
------------------------------------------------------------------------------------------------------------------
```

Yay!