Thomas Haddy
COM S 352
Due April 26, 2019

Assignment 8

1. If the time for the CPU to execute a calculation is equal to the I/O time completion, the processor and peripheral device running at the same time will take half the time than running them one at a time i.e. buffer vs. no buffer.
   Let C be the calculation time for the program
   Let I be the total I/O time
   Therefore, the best running time with a buffer is the max(C, I) and the best without a buffer is C + I
   So, $[(C\ +\ I)/2 \leq max(C, I) \leq (C + I)]$
   The values will be between (C + I)/2 and (C + I) making the running time by at most a factor of two.

2.
   a. Since the data is updated infrequently, there is less need of keeping track of the indexes. Random order accessing makes index and hash file organization the best for frequency.
      Therefore, hashed and indexed file organizations will maximize efficiency in terms of speed of access, use of storage space, and ease of updating.

   b. Since the data is accessed in its entirety, indexed sequential file organization will maximize efficiency in terms of speed of access, use of storage space, and ease of updating.

   c. Even with the data be updated frequently this time, the access is still in random order. That means that indexed or hashed file organization will maximize efficiency in terms of speed of access, use of storage space, and ease of updating.

3.
   a. Since there are two pointers rather than one for a single process, that processes priority will increase for the ready queue.

   b. The advantage is that important or critical processes will be given a higher priority for this scheme.

c. Give more time to processes that should have higher priority by including more time quantums for a higher priority process.

4. For FCFS, the order is determined by the arrival time. The first process to arrive will be served.
   For Non-preemptive, the order is determined by the priority number. The smaller the number, the higher priority if there are two or more processes waiting.
   For Round-Robin, the order is determined by arrival time, but the time quantum allows it to run for a fixed time before switching to another.
   a.

FCFS		Non-preemptive		Round Robin (q = 30)	
Time (ms)	Process Running	Time (ms)	Process Running	Time (ms)	Process Running
0	P1	0	P1	0	P1
10	P1	10	P1	10	P1
20	P1	20	P1	20	P1
30	P1	30	P1	30	P2
40	P1	40	P1	40	P2
50	P2	50	P2	50	P1
60	P2	60	P2	60	P1
70	P3	70	P4	70	P3
80	P3	80	P4	80	P3
90	P3	90	P4	90	P3
100	P3	100	P4	100	P4
110	P3	110	P3	110	P4
120	P3	120	P3	120	P4
130	P3	130	P3	130	P3
140	P3	140	P3	140	P3

150	P3	150	P3	150	P3
160	P3	160	P3	160	P4
170	P4	170	P3	170	P3
180	P4	180	P3	180	P3
190	P4	190	P3	190	P3
200	P4	200	P3	200	P3

b. Waiting time = Finish time - (Arrival time + Burst time)
Average waiting time = Sum of waiting time for each process / total # of processes

FCFS:
$P1_{WT} = 50 - (0 + 50) = 0$
$P2_{WT} = 70 - (20 + 20) = 30$
$P3_{WT} = 170 - (40 + 100) = 30$
$P4_{WT} = 210 - (60 + 40) = 110$

$FCFS_{Avg} = (0 + 30 + 30 + 110) / 4 = 42.5\ ms$

Non-preemptive:
$P1_{WT} = 50 - (0 + 50) = 0$
$P2_{WT} = 70 - (20 + 20) = 30$
$P3_{WT} = 210 - (40 + 100) = 70$
$P4_{WT} = 110 - (60 + 40) = 10$

$NP_{Avg} = (0 + 30 + 70 + 10) / 4 = 27.5\ ms$

Round Robin:
$P1_{WT} = 70 - (0 + 50) = 20$
$P2_{WT} = 50 - (20 + 20) = 10$
$P3_{WT} = 210 - (40 + 100) = 70$
$P4_{WT} = 170 - (60 + 40) = 70$

$RR_{Avg} = (20 + 10 + 70 + 70) / 4 = 42.5\ ms$

c. Turnaround time = Finish time - Arrival time
Average turnaround time = Sum of turnaround time for each process / total # of processes

FCFS:
$P1_{TT} = 50 - 0 = 50$
$P2_{TT} = 70 - 20 = 50$
$P3_{TT} = 170 - 40 = 130$
$P4_{TT} = 210 - 60 = 150$

$FCFS_{Avg} = (50 + 50 + 130 + 150) / 4 = 95\ ms$

Non-preemptive:
$P1_{TT} = 50 - 0 = 50$
$P2_{TT} = 70 - 20 = 50$
$P3_{TT} = 210 - 40 = 170$
$P4_{TT} = 110 - 60 = 50$

$NP_{Avg} = (50 + 50 + 170 + 50) / 4 = 80\ ms$

Round Robin:
$P1_{TT} = 70 - 0 = 70$
$P2_{TT} = 50 - 20 = 30$
$P3_{TT} = 210 - 40 = 170$
$P4_{TT} = 170 - 60 = 90$

$RR_{Avg} = (70 + 30 + 170 + 90) / 4 = 95\ ms$

5. README:
The programs both read an input file, convert the text to uppercase, and write it to the output file.
Do the following to run the 2 programs:
   1. Edit the input.txt file to your heart’s content
   2. Open terminal to where you unzipped my hw
   3. Type ‘make’
   4. Execute ‘./a’

5. Type ‘input.txt’ for the first prompt
6. Type ‘output.txt’ for the second prompt
7. Look at ‘output.txt’, then delete its contents and save it
8. Execute ‘./b’
9. Type ‘input.txt’ for the first prompt
10. Type ‘output.txt’ for the second prompt
11. Type ‘make clean’ when finished

Note
You can type ‘input.txt’ for input file and ‘output.txt’ for the output file
from the user input. The path starts at your terminal’s current directory so this should work assuming you are in the directory where you unzipped my hw.