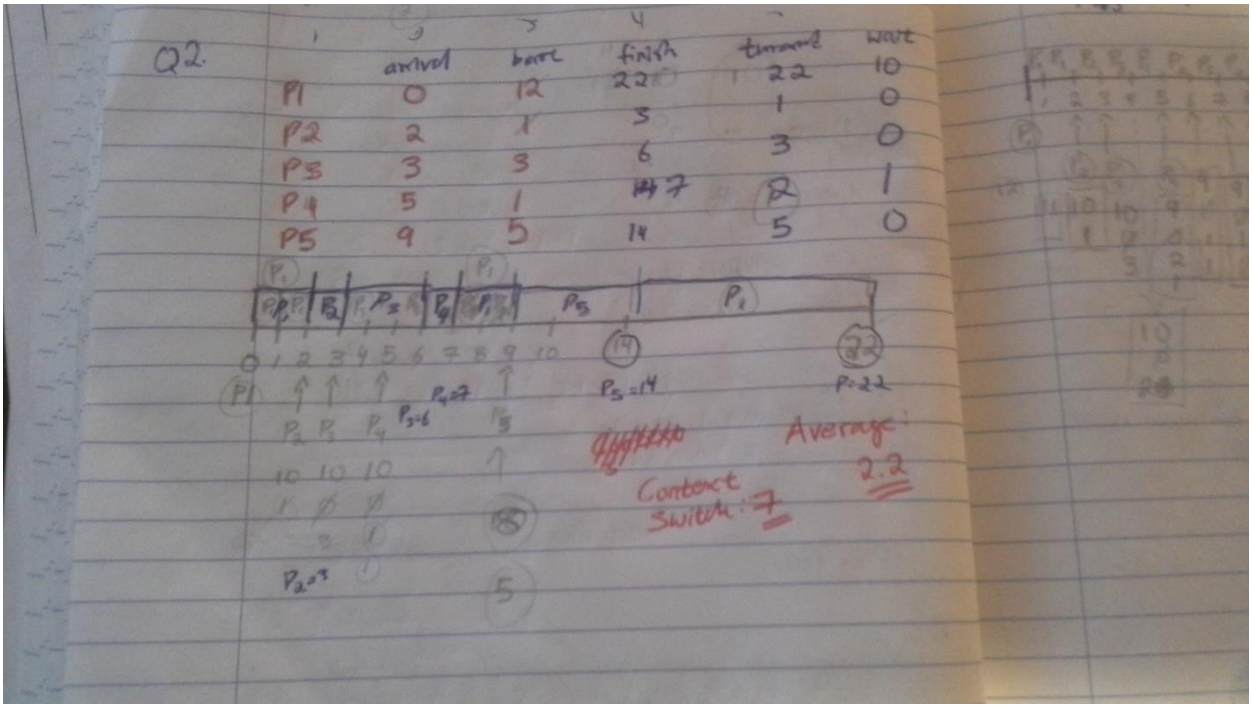
CPSC 457 Assignment 3

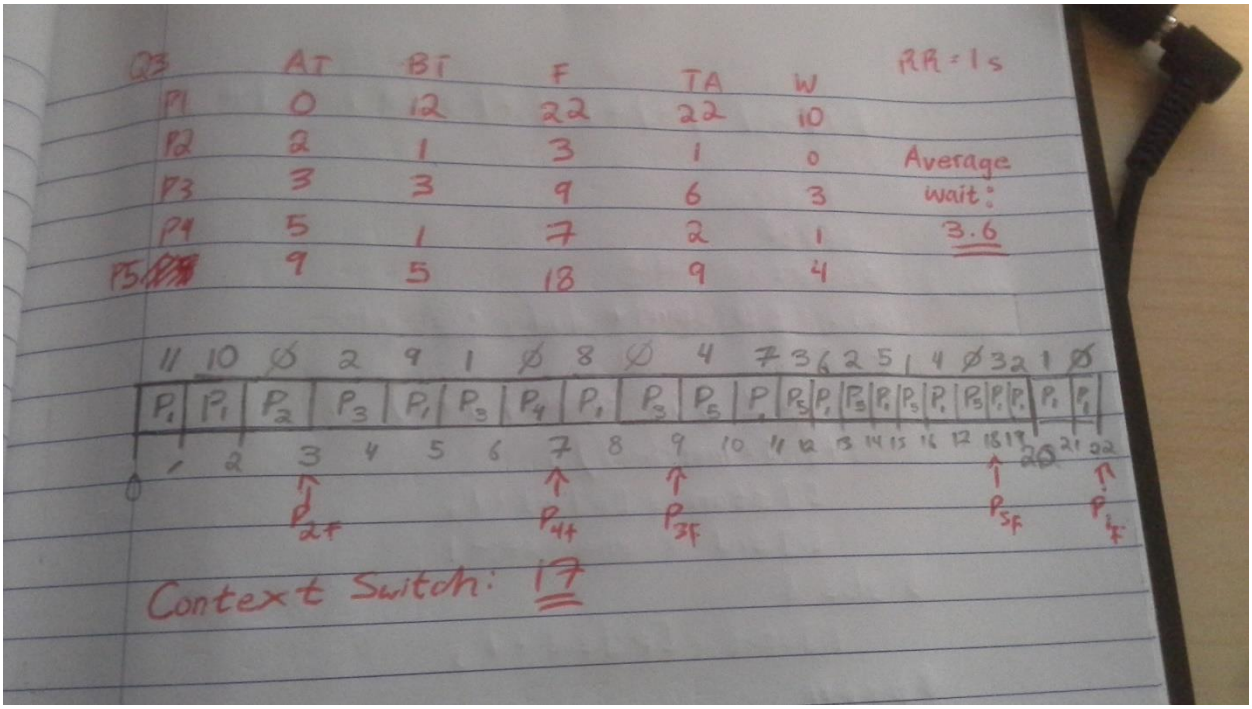
Question 1:

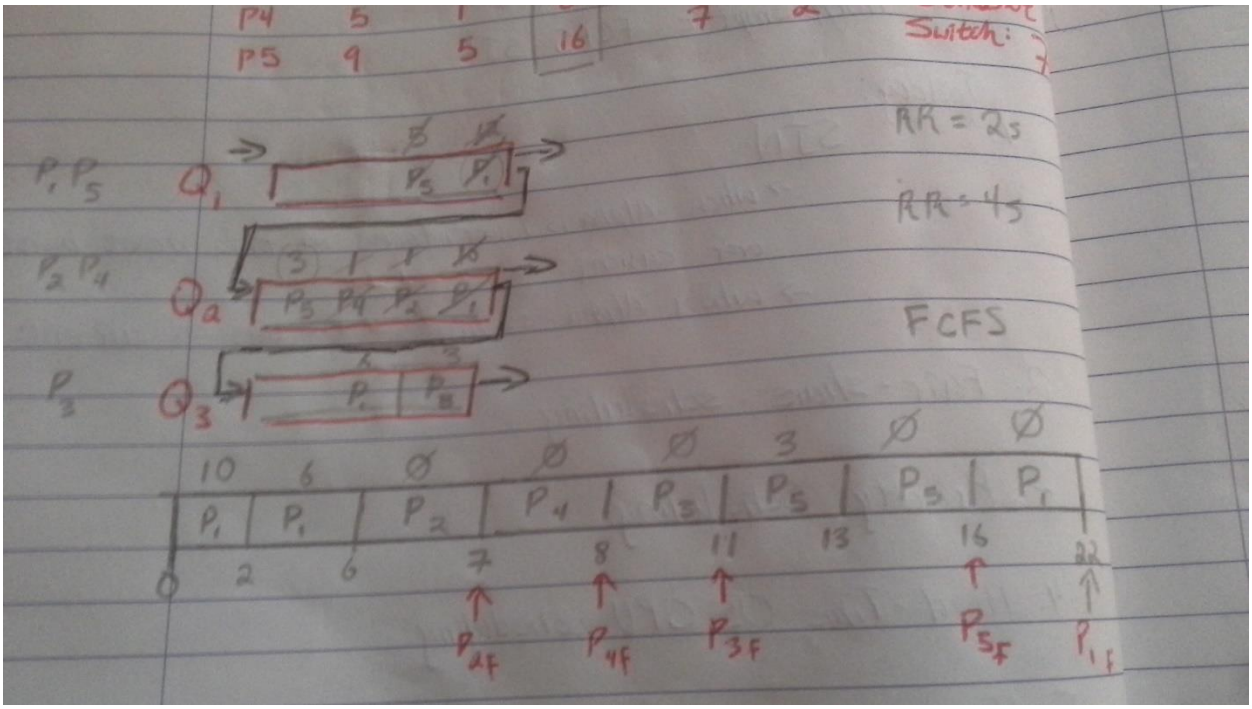
1 – (0.75)^8 = 0.89988

Question 2:

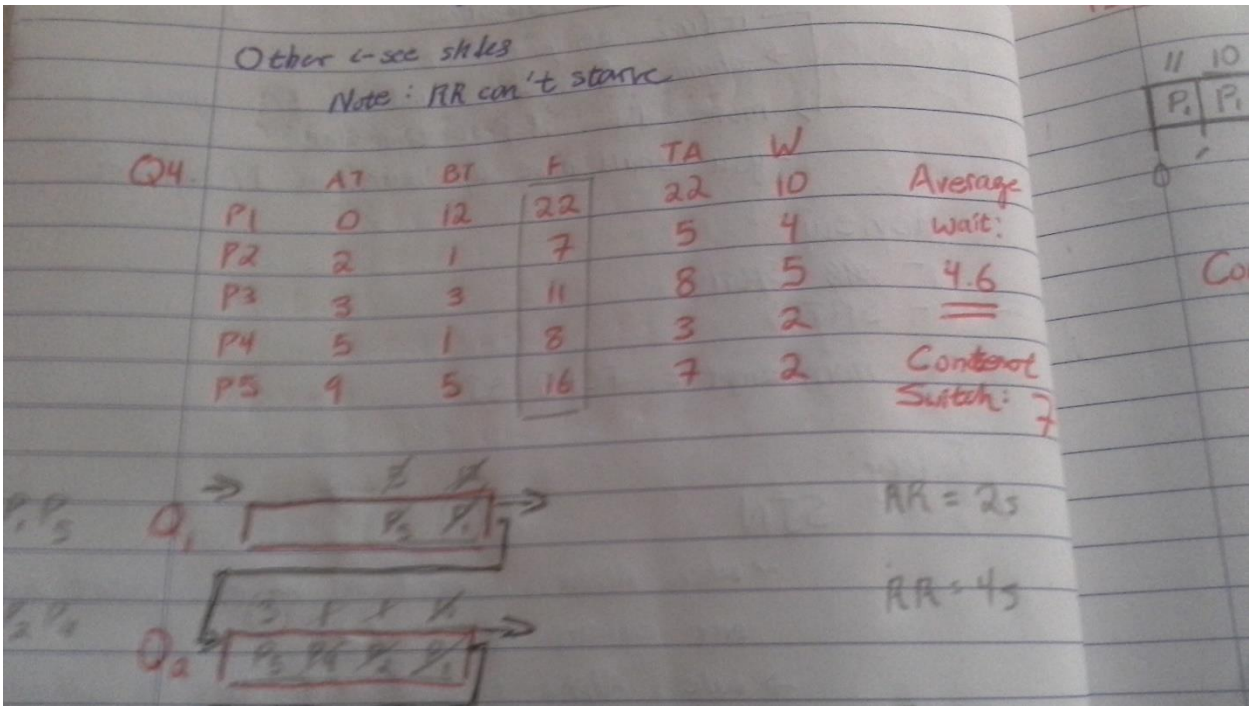


Question 3:




P4 5 1 16 7 2
P5 9 5
Switch: 7
RR = 2s
RR = 4s
FCFS
Q1
Q2
Q3
10 6 Ø Ø Ø 3 Ø Ø
P1 P1 P2 P4 P3 P5 P5 P1
0 2 6 7 8 11 13 16 22
P4F P4F P3F P5F P1F

Question 4:


Other case see slides
Note: RR can't start
Q4
AT BT F TA W
P1 0 12 22 22 10
P2 2 1 7 5 4
P3 3 3 11 8 5
P4 5 1 8 3 2
P5 9 5 16 7 2
Average wait: 4.6
Context Switch: 7
RR = 2s
RR = 4s
Q1
Q2

Question 6:

Medium.txt

# threads	Observed timing	Observed speedup compared to original	Expected speedup
Original program	0m32.265s	1.0	1.0
1	0m31.764s	~1.03	1.0
2	0m17.800s	~1.88	2.0
3	0m13.857s	~2.46	3.0
4	0m12.034s	~2.66	4.0
8	0m8.344s	~4.00	8.0
16	0m7.302s	~4.57	16.0

Hard.txt

# threads	Observed timing	Observed speedup compared to original	Expected speedup
Original program	0m10.932s	1.0	1.0
1	0m10.986s	~1.00	1.0
2	0m11.083s	~0.90	2.0
3	0m11.173s	~0.90	3.0
4	0m11.216s	~0.90	4.0
8	0m10.994s	~1.00	8.0
16	0m10.920s	~1.00	16.0

Hard2.txt

# threads	Observed timing	Observed speedup compared to original	Expected speedup
Original program	0m11.165s	1.0	1.0
1	0m11.120s	~1.00	1.0
2	0m11.708s	~1.00	2.0
3	0m11.128s	~1.00	3.0
4	0m11.190s	~1.00	4.0
8	0m11.369s	~1.00	8.0
16	0m10.851s	~1.10	16.0

Comment:

Unfortunately, as shown in the hard texts, it's clear that my code isn't nearly as efficient as I would have thought it to be. To speculate, I believe the reason why the medium.txt runs decently is that despite all the numbers being very big, they are ALL very big, as a result when I use my "divider" and slip them into threads, I can generate threads which are all roughly doing an equal amount of work. The same can't be said for hard and hard2.txt since they only carry one exponentially large value, which results in a lot of unless waiting times.