Jay Patel

CS 352

HW3

Professor Dr. Johnny Wong

4.8: Which of the following components of program state are shared across threads in a multithreaded process?

Answer: The thread of a multithread process share heap memory and global variables. Each thread has its separate set of registers values and a separate stack.

4.11: Is it possible to have concurrency but not parallelism? Explain.

Answer: It is possible to have concurrent but not parallelism because concurrent means that more than one process or thread is progressing at the same time but it does not imply that the processed are running simultaneously. The scheduling of tasks allows for concurrency, but parallelism is supported only on systems with more than one processing core.

4.17:

Answer: (Line C) Child = 5

(Line P) Parent = 0

6.14:

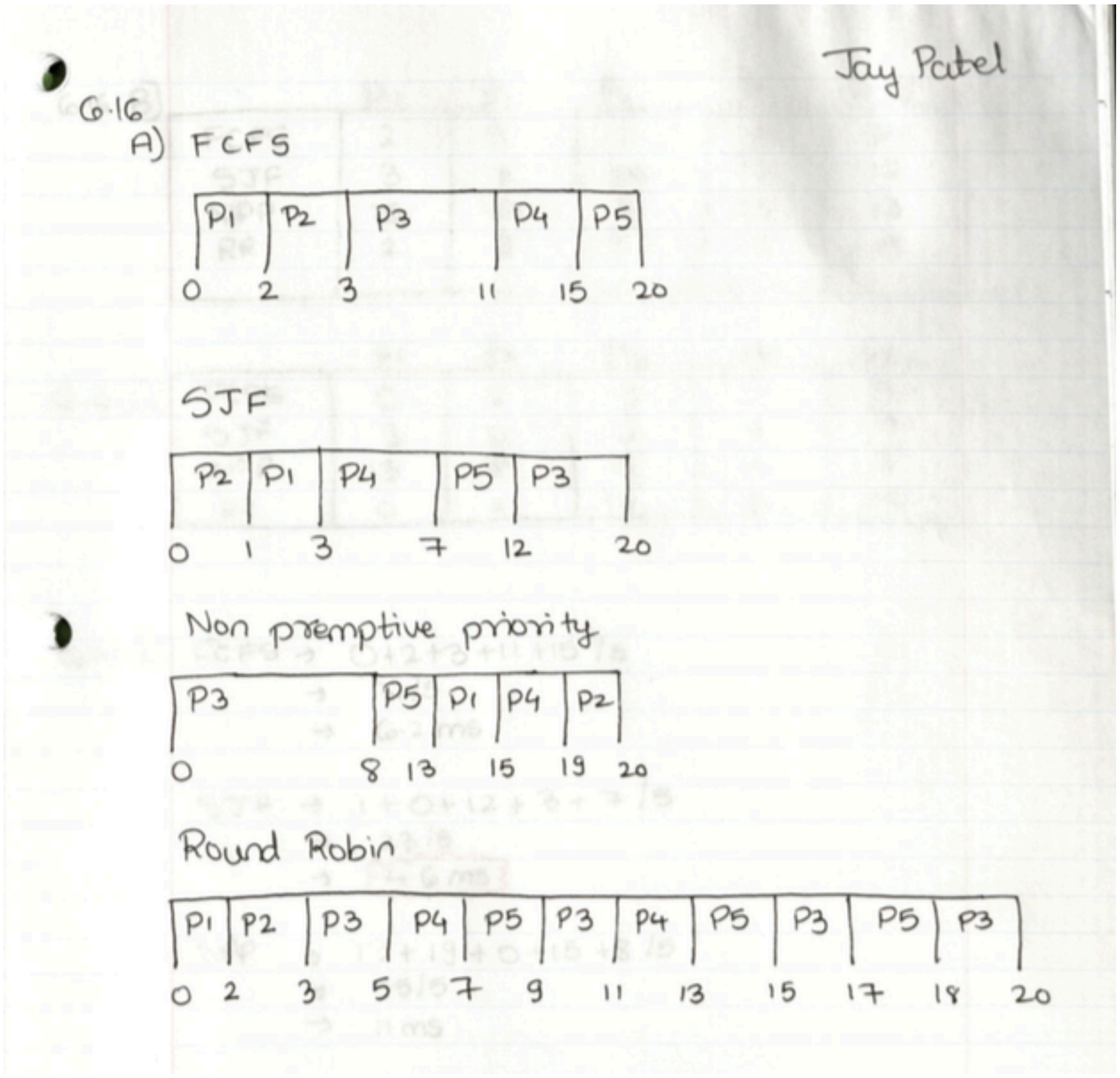
Answer:

a) The formula always makes a prediction of 100ms for the next CPU burst when a= 100 and τ0 = 100ms.
b) The most recent behavior of the process is given a much higher weight than the previous history related with the process. Similarly, the algorithm for scheduling is almost memory less and simply detects the length of the previous burst for the next quantum of CPU execution when a= 0.99 and τ0 = 10ms

6.16

Answer continue:

A)



Jay Patel

6.16

A) FCFS

P1	P2	P3	P4	P5

0 2 3 11 15 20

SJF

P2	P1	P4	P5	P3

0 1 3 7 12 20

Non premptive priority

P3	P5	P1	P4	P2

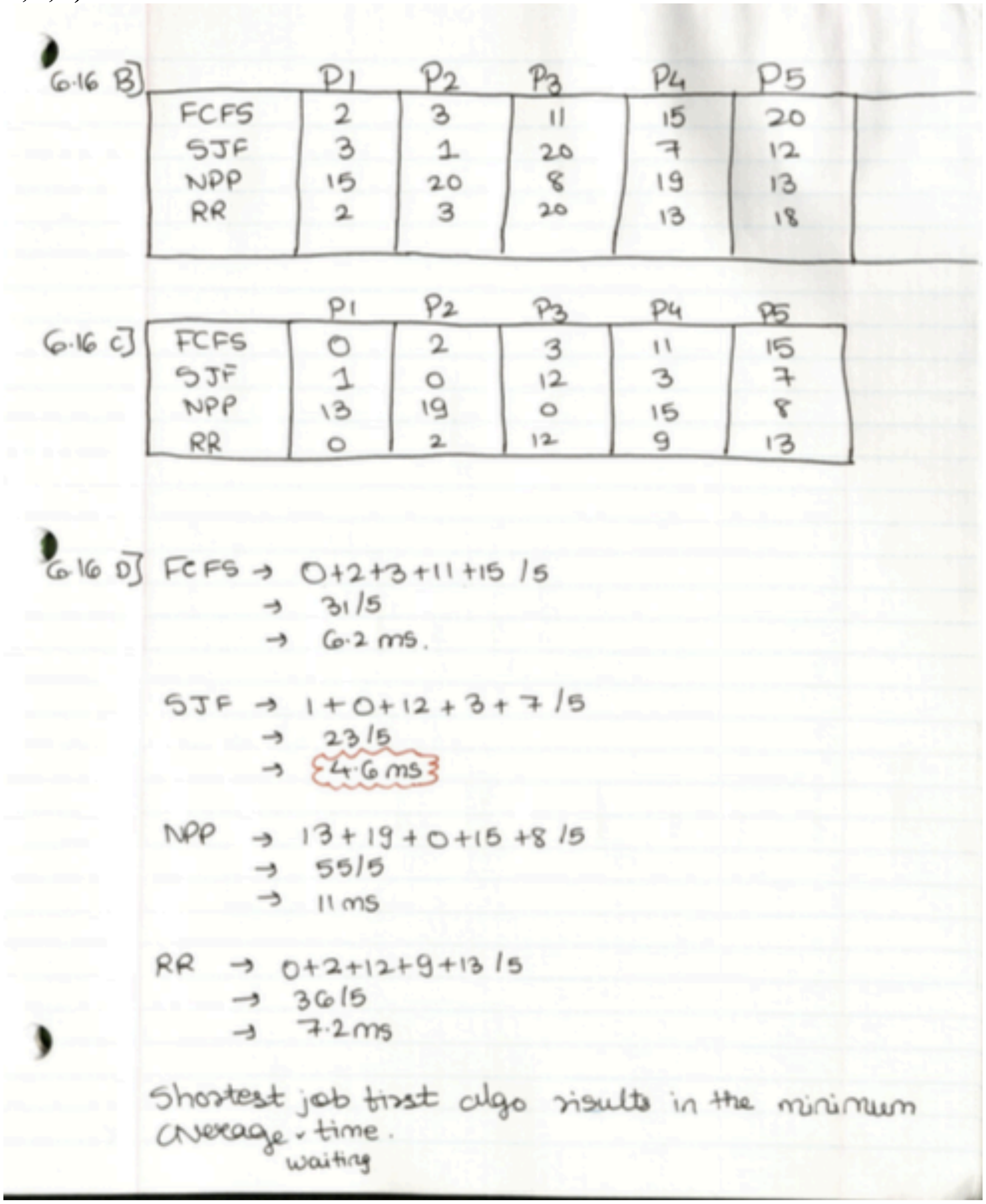
0 8 13 15 19 20

Round Robin

P1	P2	P3	P4	P5	P3	P4	P5	P3	P5	P3

0 2 3 5 7 9 11 13 15 17 18 20

B,C,D)



6.16 B]

	P1	P2	P3	P4	P5
FCFS	2	3	11	15	20
SJF	3	1	20	7	12
NPP	15	20	8	19	13
RR	2	3	20	13	18

6.16 C]

	P1	P2	P3	P4	P5
FCFS	0	2	3	11	15
SJF	1	0	12	3	7
NPP	13	19	0	15	8
RR	0	2	12	9	13

6.16 D] FCFS → 0+2+3+11+15 /5
→ 31/5
→ 6.2 ms.

SJF → 1+0+12+3+7 /5
→ 23/5
→ $\boxed{4.6\text{ ms}}$

NPP → 13+19+0+15+8 /5
→ 55/5
→ 11 ms

RR → 0+2+12+9+13 /5
→ 36/5
→ 7.2 ms

Shortest job first algo results in the minimum average waiting time.

6.19:

Answer: Shortest job first and priority-based scheduling algorithms results in starvation.

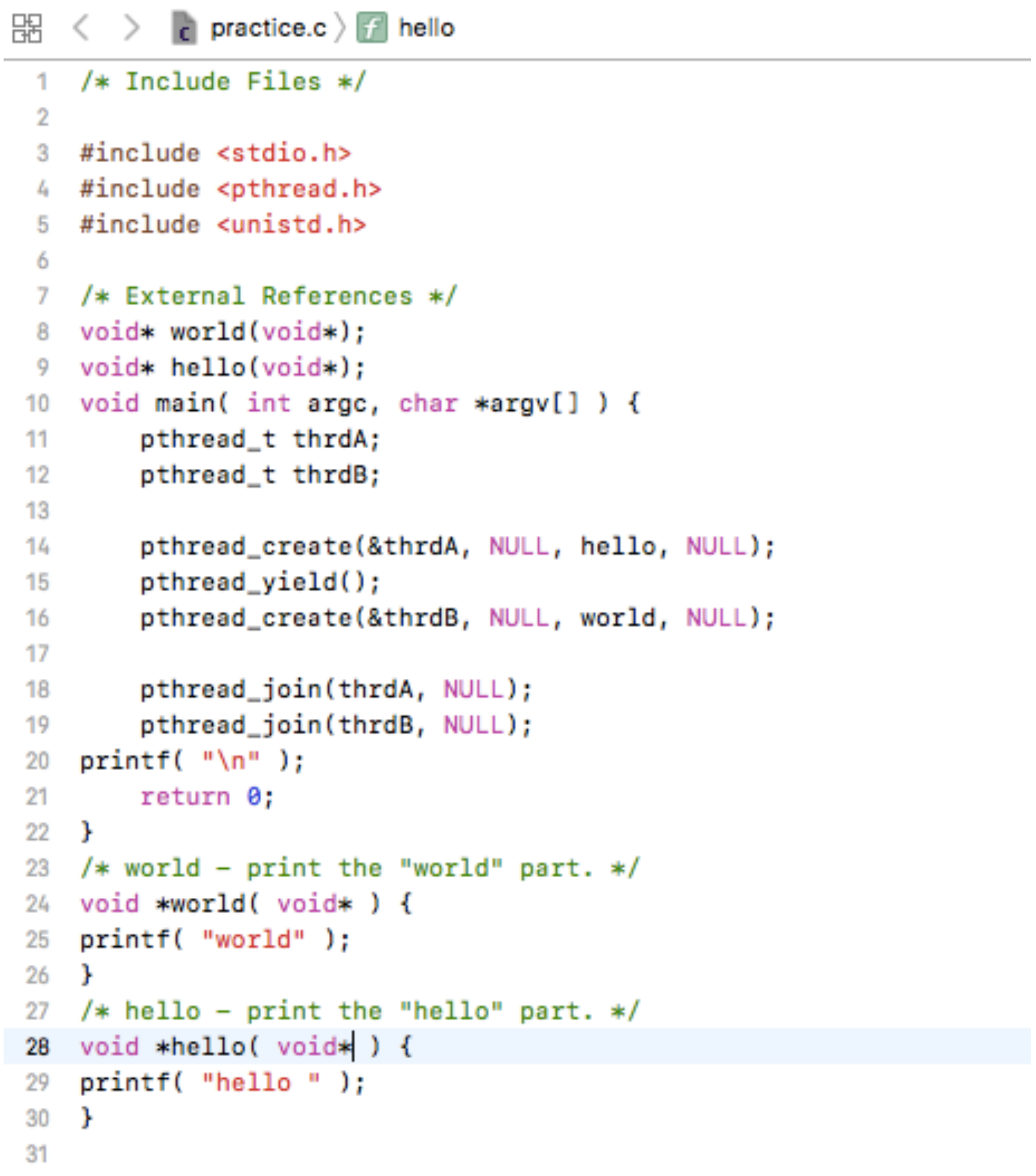
6.23:

Answer:

a) Any processes that have been in the system have higher priority because the processes start at 0 and hence new processes go to the back of the queue. The priority keeps increasing at the rate of the beta when a process runs, which is more of an increase than for the processes in the ready queue. It goes to the front of the ready queue and gets dispatched every time the process has time run out.
b) Any new processes entering the system have higher priority than any old ones because the priority starts at zero and then become negative when the process waits or run. Thus, new processes go in at the front of the queue. The priority decreases when the process runs or waits, with the waiting process decreasing faster than the running process. This is a last in first out algorithm that I studied in 228.

7:

Answer:


practice.c 〉 hello

```
/* Include Files */

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

/* External References */
void* world(void*);
void* hello(void*);
void main( int argc, char *argv[] ) {
    pthread_t thrdA;
    pthread_t thrdB;

    pthread_create(&thrdA, NULL, hello, NULL);
    pthread_yield();
    pthread_create(&thrdB, NULL, world, NULL);

    pthread_join(thrdA, NULL);
    pthread_join(thrdB, NULL);
printf( "\n" );
    return 0;
}
/* world - print the "world" part. */
void *world( void* ) {
printf( "world" );
}
/* hello - print the "hello" part. */
void *hello( void* ) {
printf( "hello " );
}
```