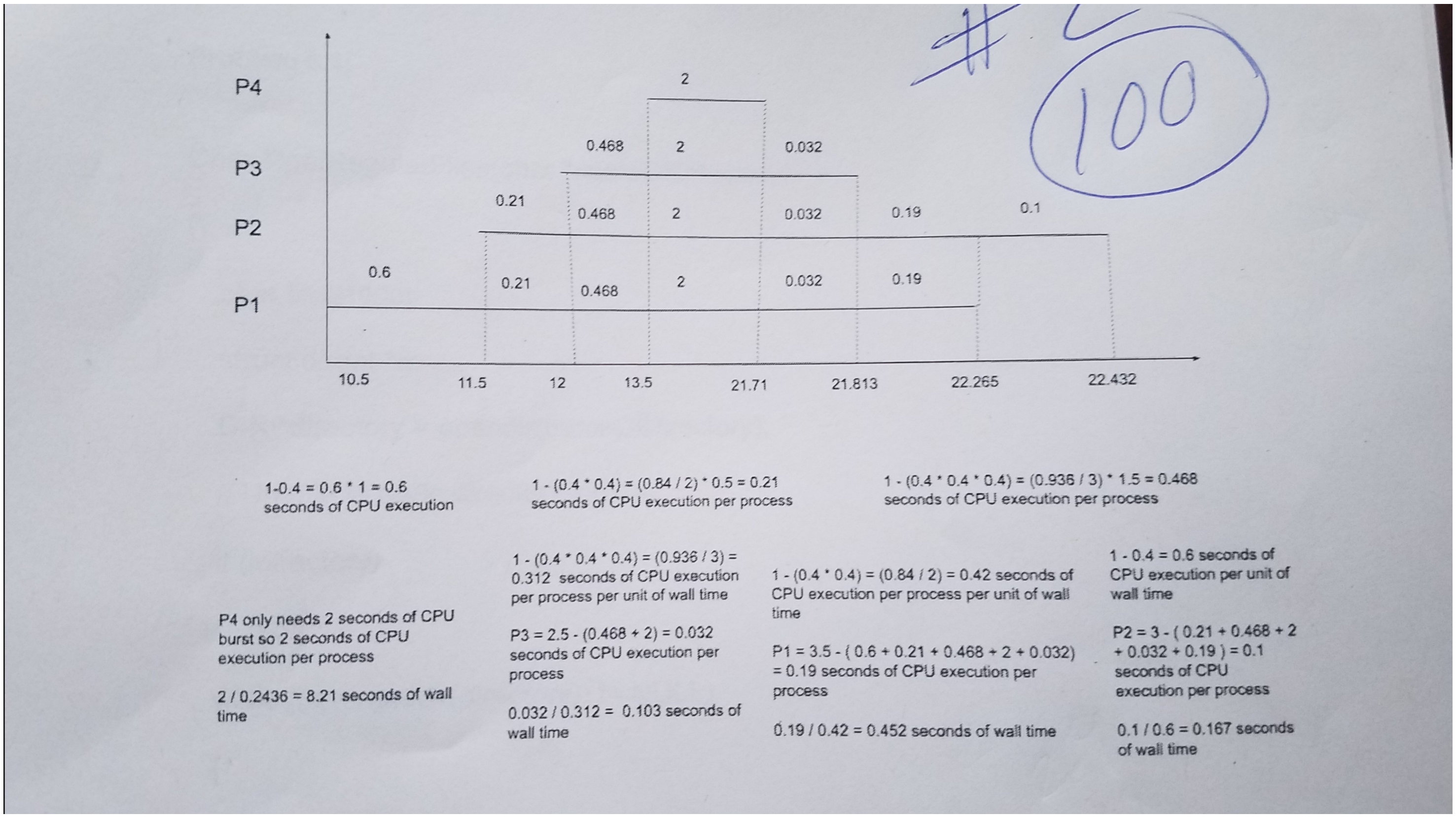
Midterm Answers

Wednesday, May 5, 2021 8:28 PM


#
100
P4
P3
P2
P1
2
0.468
2
0.032
0.21
0.468
2
0.032
0.19
0.1
0.6
0.21
0.468
2
0.032
0.19
10.5
11.5
12
13.5
21.71
21.813
22.265
22.432

1-0.4 = 0.6 * 1 = 0.6
seconds of CPU execution

1 - (0.4 * 0.4) = (0.84 / 2) * 0.5 = 0.21
seconds of CPU execution per process

1 - (0.4 * 0.4 * 0.4) = (0.936 / 3) * 1.5 = 0.468
seconds of CPU execution per process

P4 only needs 2 seconds of CPU burst so 2 seconds of CPU execution per process

2 / 0.2436 = 8.21 seconds of wall time

1 - (0.4 * 0.4 * 0.4) = (0.936 / 3) = 0.312 seconds of CPU execution per process per unit of wall time

P3 = 2.5 - (0.468 + 2) = 0.032 seconds of CPU execution per process

0.032 / 0.312 = 0.103 seconds of wall time

1 - (0.4 * 0.4) = (0.84 / 2) = 0.42 seconds of CPU execution per process per unit of wall time

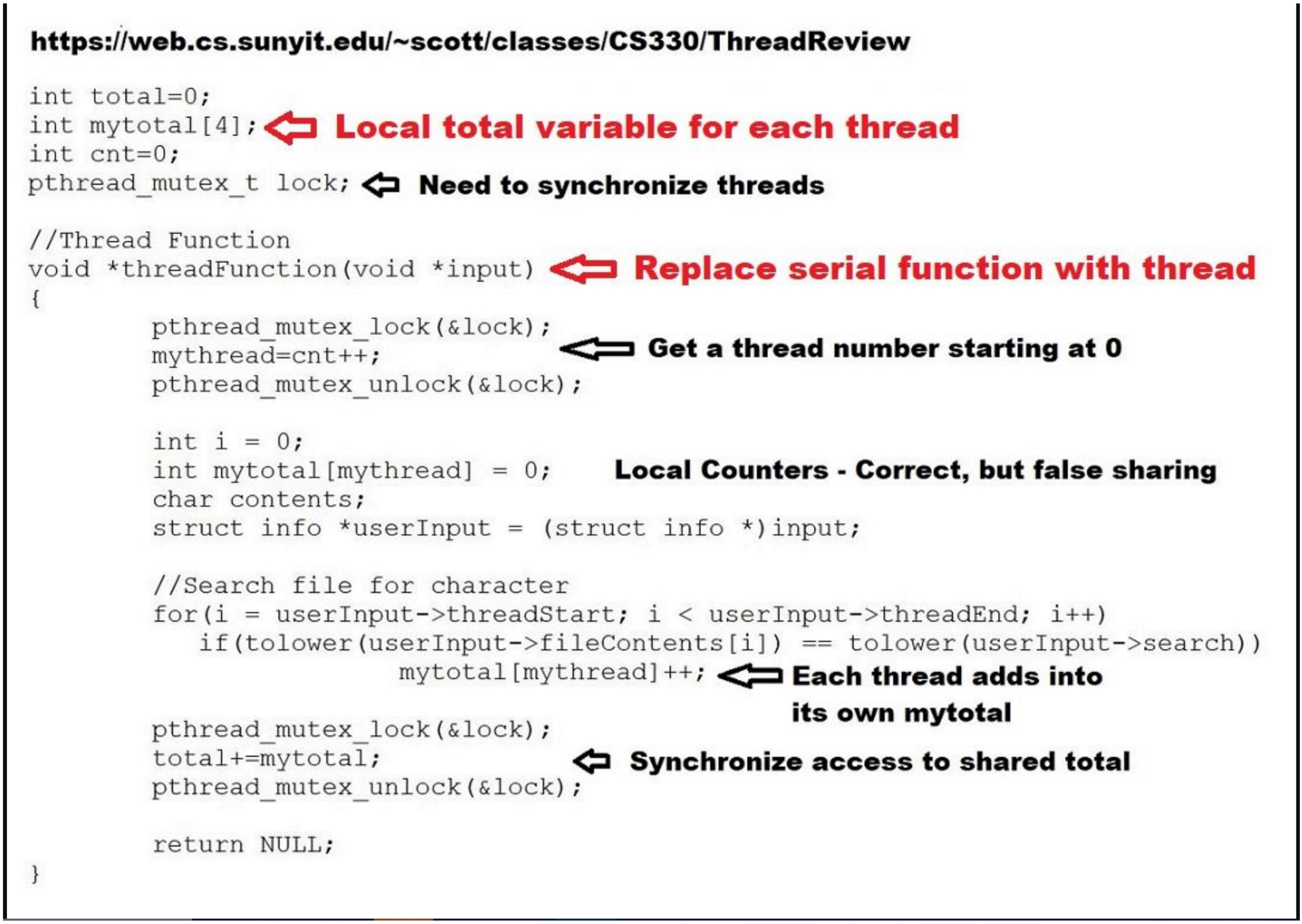
P1 = 3.5 - (0.6 + 0.21 + 0.468 + 2 + 0.032) = 0.19 seconds of CPU execution per process

0.19 / 0.42 = 0.452 seconds of wall time

1 - 0.4 = 0.6 seconds of CPU execution per unit of wall time

P2 = 3 - (0.21 + 0.468 + 2 + 0.032 + 0.19) = 0.1 seconds of CPU execution per process

0.1 / 0.6 = 0.167 seconds of wall time



https://web.cs.sunyit.edu/~scott/classes/CS330/ThreadReview

```
int total=0;
int mytotal[4]; ⇦ Local total variable for each thread
int cnt=0;
pthread_mutex_t lock; ⇦ Need to synchronize threads

//Thread Function
void *threadFunction(void *input) ⇦ Replace serial function with thread
{
        pthread_mutex_lock(&lock);
        mythread=cnt++;                 ⇦ Get a thread number starting at 0
        pthread_mutex_unlock(&lock);

        int i = 0;
        int mytotal[mythread] = 0;      Local Counters - Correct, but false sharing
        char contents;
        struct info *userInput = (struct info *)input;

        //Search file for character
        for(i = userInput->threadStart; i < userInput->threadEnd; i++)
           if(tolower(userInput->fileContents[i]) == tolower(userInput->search))
                        mytotal[mythread]++; ⇦ Each thread adds into
                                               its own mytotal
        pthread_mutex_lock(&lock);
        total+=mytotal;                 ⇦ Synchronize access to shared total
        pthread_mutex_unlock(&lock);

        return NULL;
}
```