CS3360: Homework #5

Due on April, 7 , 2023 @ 11:59 pm (firm – submission on Canvas closes 11:59 PM)

Mina Guirguis

PLEASE READ: You may discuss this problem set with other students as long as you do not share/verify answers. However, you must also write the names of other students you discussed any problem with. You must *write up your answers on your own showing your work*. Each problem has a different weight. Please state any assumptions you are making in solving a given problem. Late assignments will not be accepted with prior arrangements. By submitting this assignment, you acknowledge that you have read the course syllabus and abiding to the copyright (e.g., no posting) and Honor Code (e.g., not using note sharing sites such as Chegg, OneClass, CourseHero, etc.) requirements.

Problem 1

Consider the following real-time periodic processes:

Process	Service time	Period
A	1	8
B	2	10
C	4	16
D	4	20

Answer the following questions [**5 pts each**]:

(a) Can we guarantee that the above processes would meet their deadline under RMS?

(b) Show the execution of the above tasks under RMS until time 80 or until a process misses its deadline.

(c) Show the execution of the above tasks under EDF until time 80 or until a process misses its deadline.

(d) Suppose an additional process E requests 8 service time units with a period of 40, should this process be admitted under RMS? Explain your answer.

Problem 2

In this problem, you are asked to use OpenMP to compute the speedup you would gain from using 2, 4, 8 and 16 CPUs instead of 1 CPU. You would need to use one of the following machines to complete this problem: columbia, constitution, endeavour, enterprise, discovery, and pathfinder. You would need to access those machines through eros or zeus if you are off campus.

You are asked to generate an array of 1 million integers picked at random using the rand() function. Each element in the array is to be processed 1000 times in which we compute the modulus (remainder from dividing the random number by the integers 1 through 1000 (i.e., if x is the random number, we want to compute x % 1 through x % 1000). Your program should not output anything to the terminal in the parallel portion of it. Use timeval/gettimeofday() to record the time and compute the elapsed time for the parallel portion. Use *omp_set_num_threads*() to set the number of threads. Each experiment should be done five times and the average should be computed under 1, 2, 4. 8 and 16 CPUs. [**30 pts**]

Problem 3

Consider the following program running on an 8-core machine in which a race condition can happen with the shared variable tally.

```
const int n = 5;
int tally;

void total() {
   int count;
   for (count = 1; count<=n; count ++)
      tally++
}

void main() {
   int tally;
   int i;
   tally = 0;
   #pragma omp parallel for
   for (i = 1; i<=8; i ++)
      total()
   print (tally);
}
```

Answer the following questions [**5 pts each**]:

(a) What is the largest possible value of the shared variable tally. Assume the cores can execute at any relative speeds and that the increment statement of tally is broken into 3 instructions (load, add, store).

(b) What is the smallest possible value of the shared variable tally. Assume the cores can execute at any relative speeds and that the increment statement of tally is broken into 3 instructions (load, add, store).