Lab 3 report ECE445M

Name: __Replace with your name(s)____

EID: __Replace with your EID(s)_____

Semester: Spring 2026
Course: ECE445M

A) ***Objectives*:**
- Extend the RTOS to include blocking, priority, and four real-time periodic tasks,
- Develop minimally invasive tools to determine performance measures,
- Record debugging/performance data and download this data to the PC.

B) ***Hardware Design:***
- MSPM0G3507 Launchpad and ECE445M sensor board.

C) ***Software Design Deliverables:***
Upload entire contents of these three folders to github
inc
RTOS_Lab3_RTOSpriority
RTOS_Labs_common

1) Drawing of your data structure(s) used to implement priority (Preparation 1)
<add here>

2) Drawing of data structure(s) used to implement periodic background threads (Preparation 2)
<add here>

3) Drawing of data structure(s) used to implement blocking (Preparation 4)
<add here>

4) Drawing of possible data structure(s) if there were 100 of threads (Preparations 1 and 4)
<add here>

5) Drawing of possible data structure(s) used for two periodic background threads (Prep 2)
<add here>

D) ***Measurement Data:***
1) Plot/photo of the logic analyzer running *Testmain3*, like Figure 3.1
<add here>

2) Plot/photo of the logic analyzer running *Testmain5*, showing the time between **OS_bSignal(&Readye);** and **OS_bWait(&Readye);**
<add here>

3) Plot/photo of the logic analyzer running *Testmain6*, like Figure 3.2
<add here>

4) Table like Table 3.1 each showing performance of *Testmain6*
<add here>

5) FilterWork, MaxJitter3, NumCreated, ChecksWork, and DataLost showing performance of *realmain*
<add here>

6) Plot/photo of the logic analyzer running *TestmainCS*, like Figure 3.3, calculate thread switch time
<add here>

E) ***Analysis and Discussion Questions:***
1) While running *Testmain1* and *Testmain2*, why are Thread2 and Thread2b completely starved? How could you have designed the OS so these threads run occasionally while still preserving the priority?
<add here>

2) While running *Testmain6* list the major sources of jitter.)
<add here>

3) Why are there so many (11) calls to a random number generator in the test code?
<add here>

4) What happens to your OS if all the threads are blocked? If your OS would crash, describe exactly what the OS does? What happens to your OS if all the threads are sleeping? If your OS would crash, describe exactly what the OS does? If you answered crash to either or both, explain how you might change your OS to prevent the crash.
<add here>

5) What happens to your OS if one of the foreground threads returns? E.g., what if you added this foreground

```
void BadThread(void){ int i;
  for(i=0; i<100; i++){};
  return;
}
```

What should your OS have done in this case? Do not implement it, rather, with one sentence, say what the OS should have done? Hint: I asked this question on an exam.
<add here>

6) What are the advantages of spinlock semaphores over blocking semaphores? What are the advantages of blocking semaphores over spinlock?
<add here>

7) When running *realmain*, why the lab3 jitter is worse than the lab2 jitter?
<add here>