Electrical, Computer & Energy Engineering
UNIVERSITY OF COLORADO **BOULDER**

ECEN 5623 Exercise #2 Service Scheduling Feasibility

Course Name: **Real-Time Embedded Systems**
Submission Date: **RESUBMISSION [5th Aug, 2019] --- Originally submitted on June 21st, 2019**
Student Name: **Rushi James Macwan**
Board Used: Raspberry Pi 3B+

Note: Correct answer is in Blue Font

Credits: *All codes utilized for completing this assignment have been reused and modified accordingly. The codes are originally authored by* ***Dr. Siewert*** *and is* ***NOT*** *my original work. All code fragments have been reused and duly credited here for its reuse. Appropriate code explanations have been made wherever required.*

1) [5 points] If you're using embedded Linux, make yourself an account on your R-Pi3b, Jetson Nano, or Jetson TK1 system. To do this on Jetson TKI, use the reset button if the system is locked, use the well-known "ubuntu" password to login, and then use "sudo adduser", enter the well-known password, and enter user information as you see fit. Add your new user account as a "sudoer" using "visudo" right below root with the same privileges (if you need help with "vi", here's a quick reference or reference card– use arrows to position cursor, below root hit Esc, "i" for insert, type username and privileges as above, and when done, Esc, ":", "wq"). The old unix vi editor was one of the first full-screen visual editors – it still has the advantage of being found on virtually any Unix system in existence, but is otherwise cryptic – along with Emacs it is still widely used in IT, by developers and systems engineers, so it's good to know the basics. If you really don't like vi or Emacs, your next best bet is "nano" for Unix systems. Do a quick "sudo whoami" to demonstrate success. Logout of ubuntu and test your login, then logout. Use Alt+Print-Screen to capture your desktop and save as proof you set up your account. Note that you can always get a terminal with Ctrl+Alt+t key combination. If you don't like the desktop, you can try "GNOME Flashback" and please play around with customizing your account as you wish. Make sure you can access our class web page on Firefox (or default browser) and set your home page to http://ecee.colorado.edu/~ecen5623/index_summer.html . Overall, make sure you are comfortable with development, debug, compiler general native or cross-development tools and document and demonstrate that you know them.

Sol.

I have performed a complete setup of a Raspberry Pi 3B+ board. Please, refer to the two screenshots provided in the attached zipped file (named: Exercise-2 Attachments) with this submission under the folder name: Problem-1.

Further information on my setup is as under:

1. The attached screenshot (named **Setup-1**) showcases the login page on the R-Pi 3B+ platform.
2. The attached screenshot (named **Setup-2**) portrays the command line interface on the R-Pi 3B+ platform. The instructions on the command line interface showcase that the user is having the root privileges and provides information on the last login on the device. Also, a switch to the root configuration and a return to the main user is also performed.
3. The attached screenshot (named **Setup-3**) showcases the use of the GNU nano 2.7.4 text editor available on the R-Pi 3B+ platform.
4. The attached screenshot (named **Setup-4**) showcases an active internet connection and the course webpage available on the Chrome Web Browser.

2) [10 points] Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" [available on Canvas], by Gene Carlow and provide an explanation and critique of the frequency executive architecture. What advantages and disadvantages does the frequency executive have compared to the real-time threading and tasking implementation methods for real-time software systems? Please be specific about the advantages and disadvantages and provide at least 3 advantages as well as 3 disadvantages.

Sol.

Overall understanding of Paper:

The **frequency executive architecture** is primarily focused on the usage of static scheduling mechanism which is computed offline and stored in a scheduling table. It forms a way to sequence tasks for real-time systems where task scheduling is non pre-emptive and thus no real-time operating system is required. Thus, critical system functions that are required to service repeatedly and deterministically at every certain point of instance (periodically) relative to the beginning of the system execution starting cycle or loop can be serviced through the frequency executive architecture. This essentially applies to the flight application software tasks like the GN&C, SM, and VCO routines which are required to interact with the flight crew periodically and in real-time.

Summary:

The brief note on the paper can be summarized to establish that the PASS is designed to support various critical avionics system environments and time-critical applications that are based on hard real-time deadlines. The frequency executive architecture as used by the application software of the PASS as supported by its flight control operating system (FCOS) uses the concept of time slicing for allocating the CPU resources. Essentially, the frequency executive architecture allocates the CPU resources to the time-critical tasks of the PASS application software at a fixed frequency which is dependent on the task priority. A high-priority task will be served much often than a low-priority task. To conclude, a frequency executive architecture eliminates a complex system and replaces it with a relatively simple and streamlined operating system that service tasks at certain intervals. It is most important to point out here that these tasks (that are serviced by the frequency executive architecture) – will also be required to service at certain intervals as they continuously provide and communicate control and data information across the PASS.

The use of these frequency executive architectures comes into play into robust hard RT-systems that deal with dynamic environments like the space satellites and spacecraft systems. These systems (i.e. the CubeSat) is consistently and continuously requiring and digesting new data from the external environment. It could be reading the position coordinates, the spacecraft attitude and determination, the level of battery power in the spacecraft power storage unit, the signals from the ground stations for necessary physical and system changes, etc. These spacecraft systems, based on their form factor and applications are required to be continuously monitored and/or directed by the Operating System on board the spacecraft and often through the ground stations. Frequency executive architecture helps to service those critical tasks without extraordinary system scheduling complications at regular intervals such that the system does not miss critical information deadlines. The frequency executive architecture often protects such highly critical and application specific systems from the failures of scheduling inefficiencies and often reduces the system processing requirements for service scheduling management as the frequency executive architecture is deterministic in nature.

More on the relevance of the frequency executive architecture can be understood from its advantages and disadvantages as presented below:

Advantages and Disadvantages:

Advantages of Frequency Executive Architecture:

1. The frequency executive architecture fairly provides a simple and deterministic design alternative for scheduling periodic system tasks.
2. It rules out the need for RTOS for periodic and highly deterministic services which are critical and needs to be serviced periodically at certain intervals relative to the system starting point.

3. Systems designed based on the frequency executive architecture can be validated and tested with a high level of certainty.
4. Since the frequency executive architecture uses non pre-emptive scheduling, it is safe from potential deadlocks, race conditions, or blocking and is therefore serviced reliably, periodically and very deterministically.
5. The task dispatching is very efficient in the frequency executive architecture.

Disadvantages of Frequency Executive Architecture:

1. The frequency executive architecture which is based on a scheduling mechanism that is computed offline and is stored in a scheduling table, there is only limited flexibility to accommodate change.
2. The tasks must have a pre-determined release time and does not fit a dynamic priority scheduling mechanism. This is because frequency executive architecture uses non pre-emptive scheduling and that every task is serviced a specific period regularly. That means, the task must have a fixed release time for deterministic and reliable execution of services.
3. Finding a proper schedule for the set of tasks may require significant offline computation to reach a reliable schedule.
4. It requires a thorough testing and debugging of tasks that can run parallel to each other to avoid potential run-time errors.
5. Segmenting tasks into smaller fragments (smaller tasks) is often difficult and is susceptible to errors in the schedule design.

Credits: I duly credit Gene Carlow's work in his research paper for the above answer. In addition to that, I duly credit the external source that I used for framing this answer which can be accessed here.

3) [50 points] Download feasibility example code and build it on a Jetson or alternate system of your choice (or ECES Linux if you have not mastered the Jetson yet) and execute the code. Compare the tests provided to analysis using Cheddar for the first 4 examples tested by the code. Now, implement remaining examples [5 more] of your interest from those that we reviewed in class (found here). Complete analysis using Cheddar RM. In cases where RM fails, test EDF or LLF to see if it succeeds and if it does, explain why. Cheddar uses both service simulations over the LCM of the periods as well as feasibility analysis based on the RM LUB and scheduling-point/completion-test algorithms, referred to as "Worst Case Analysis". Does your modified Feasibility code agree with Cheddar analysis in all 5 additional cases? Why or why not?

Sol.

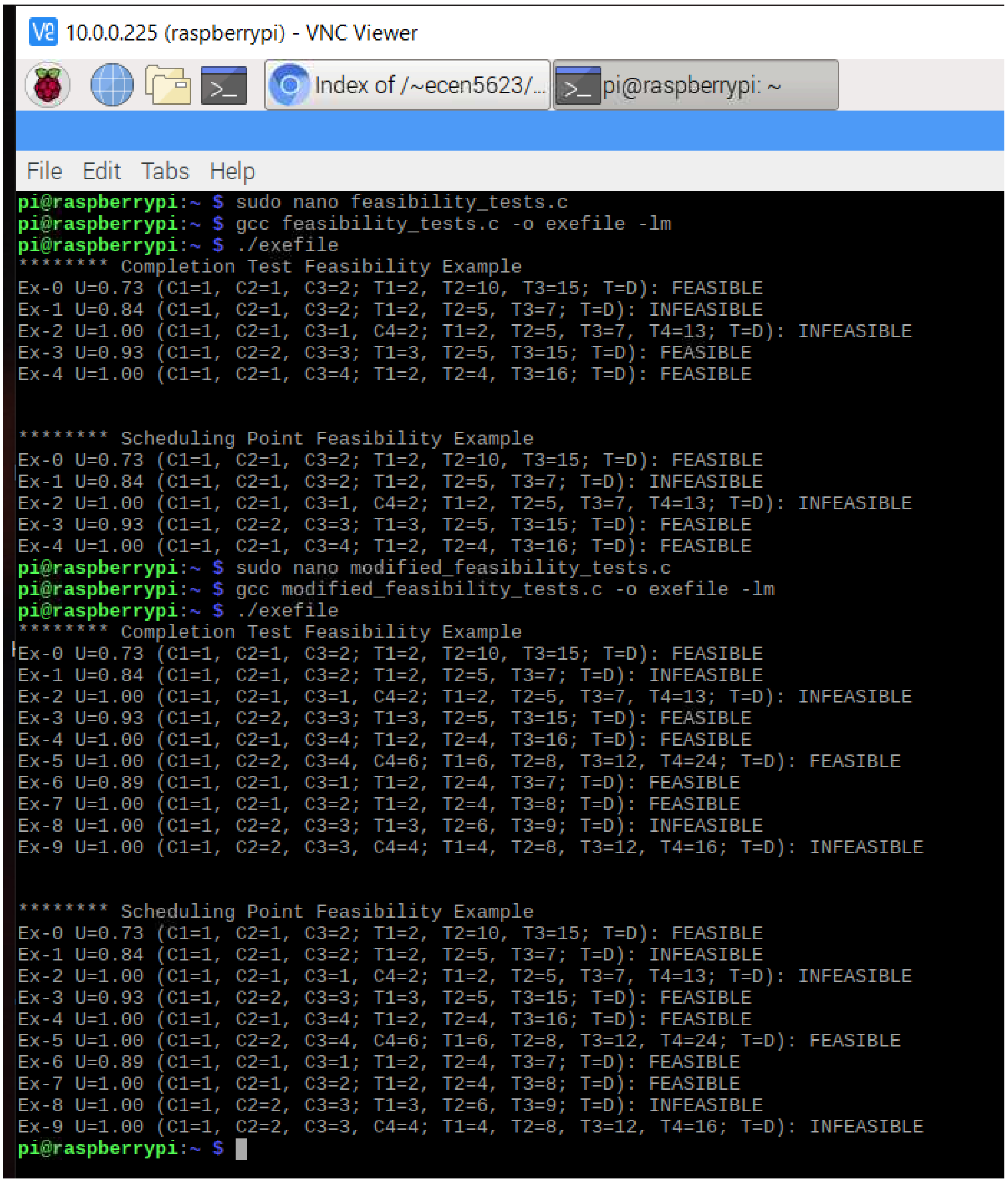
PLEASE NOTE:

All the Cheddar Analysis screenshots are provided in the attached zipped file with this submission. Also, the modified code has been also added in the attached zipped file. All the Problem-3 attachments are placed in the folder named "Problem-3" in the zipped file.

Also, for each of the ten examples (example-0 through 9) in the code, all of the three scheduling policy analysis (i.e. RM, EDF and LLF) has performed. Again, the screenshots for each of these scheduling policies for all of the ten examples have been added to the zipped file.

The feasibility example code carries out the completion test and scheduling point feasibility tests for ten examples (0 through 9) with a set of 3-4 services (S1 through S4) with different period and execution time parameters (i.e. Ti and Ci), with the service period equal to the corresponding service deadline (i.e. T = D).

Please, refer to the below attached screenshot for the output of the feasibility example code that is executed on the R-Pi 3B+ platform. The screenshot also contains a second execution of the code where five additional examples are added to the original five examples for the feasibility test.



```
pi@raspberrypi:~ $ sudo nano feasibility_tests.c
pi@raspberrypi:~ $ gcc feasibility_tests.c -o exefile -lm
pi@raspberrypi:~ $ ./exefile
******** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE


******** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
pi@raspberrypi:~ $ sudo nano modified_feasibility_tests.c
pi@raspberrypi:~ $ gcc modified_feasibility_tests.c -o exefile -lm
pi@raspberrypi:~ $ ./exefile
******** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-6 U=0.89 (C1=1, C2=1, C3=1; T1=2, T2=4, T3=7; T=D): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=1, C3=2; T1=2, T2=4, T3=8; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=3, C4=4; T1=4, T2=8, T3=12, T4=16; T=D): INFEASIBLE


******** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
Ex-6 U=0.89 (C1=1, C2=1, C3=1; T1=2, T2=4, T3=7; T=D): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=1, C3=2; T1=2, T2=4, T3=8; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
Ex-9 U=1.00 (C1=1, C2=2, C3=3, C4=4; T1=4, T2=8, T3=12, T4=16; T=D): INFEASIBLE
pi@raspberrypi:~ $
```

<u>Cheddar Analysis for each of the 10 examples:</u>

The Cheddar analysis for each of the 10 examples is as below. The screenshots for the scheduling have not been added here but they are available in the zipped file attachment. Wherever the RM analysis fails, appropriate explanation for the EDF and LLF analysis has been provided.

Example-0 Analysis:

For **Example-0**, we have the following parameters:

Example-0						
S1	**C1**	1	**T1**	2	**Utotal**	0.73
S2	**C2**	1	**T2**	10		
S3	**C3**	2	**T3**	15		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

Analysis Explanation:

In example-0, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

Example-1 Analysis:

For **Example-1**, we have the following parameters:

Example-1						
S1	**C1**	1	**T1**	2	**Utotal**	0.84
S2	**C2**	1	**T2**	5		
S3	**C3**	2	**T3**	7		

For the above service/task information, the code proves that the scheduling is **INFEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is not schedulable and therefore, **INFEASIBLE** as some service deadlines will be missed. However, the Cheddar Analysis for the EDF and LLF scheduling policies showcase that the service set is schedulable (as no deadlines were missed) and therefore, **FEASIBLE**.

Analysis Explanation:

In example-1, the task set is INFEASIBLE according to the code. The Cheddar RM analysis shows that some task deadlines will be missed and therefore the task set is NOT schedulable. This is true because the service S3 requires a capacity of 2 for a period of 7. Because of the RM scheduling policy of servicing high frequency tasks with higher priority, the service S3 with the lowest priority is not serviced before its deadline in its first cycle of a period of 7 units. The service S3 is serviced after its first deadline and therefore the task set is not schedulable. As opposed to the Cheddar RM analysis, the EDF and LLF Cheddar analysis shows that service priorities are set according to the amount of execution time left before the service deadlines are due (i.e. according to the EDF and LLF policies). Given that, the EDF and LLF Cheddar analysis allows the task set to meet ALL the deadlines and therefore the services are schedulable according to the EDF and LLF policies.

Example-2 Analysis:

For **Example-2**, we have the following parameters:

Example-2						
S1	**C1**	1	**T1**	2	**Utotal**	1.00
S2	**C2**	1	**T2**	5		
S3	**C3**	1	**T3**	7		
S4	**C4**	2	**T4**	13		

For the above service/task information, the code proves that the scheduling is **INFEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is not schedulable and therefore, **INFEASIBLE** as some service deadlines will be missed. However, the Cheddar Analysis for the EDF and LLF scheduling policies showcase that the service set is schedulable (as no deadlines were missed) and therefore, **FEASIBLE**.

<u>Analysis Explanation:</u>

In example-2, the task set is INFEASIBLE according to the code. The Cheddar RM analysis shows that some task deadlines will be missed and therefore the task set is NOT schedulable. This is true because the service S4 requires a capacity of 2 for a period of 13. Because of the RM scheduling policy of servicing high frequency tasks with higher priority, the service S4 with the lowest priority is not serviced before its deadline in its first cycle of a period of 13 units. The service S4 is serviced after its first deadline and therefore the task set is not schedulable. As opposed to the Cheddar RM analysis, the EDF and LLF Cheddar analysis shows that service priorities are set according to the amount of execution time left before the service deadlines are due (i.e. according to the EDF and LLF policies). Given that, the EDF and LLF Cheddar analysis allows the task set to meet ALL the deadlines and therefore the services are schedulable according to the EDF and LLF policies.

<u>Example-3 Analysis:</u>

For **Example-3**, we have the following parameters:

Example-3						
S1	**C1**	1	**T1**	3	**Utotal**	0.93
S2	**C2**	2	**T2**	5		
S3	**C3**	3	**T3**	15		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

<u>Analysis Explanation:</u>

In example-3, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

<u>Example-4 Analysis:</u>

For **Example-4**, we have the following parameters:

Example-4						
S1	**C1**	1	**T1**	2	**Utotal**	1.00
S2	**C2**	1	**T2**	4		
S3	**C3**	4	**T3**	16		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

Analysis Explanation:

In example-4, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

Example-5 Analysis:

For **Example-5**, we have the following parameters:

Example-5						
S1	**C1**	1	**T1**	6	**Utotal**	1.00
S2	**C2**	2	**T2**	8		
S3	**C3**	4	**T3**	12		
S4	**C4**	6	**T4**	24		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

Analysis Explanation:

In example-5, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

Example-6 Analysis:

For **Example-6**, we have the following parameters:

Example-6						
S1	**C1**	1	**T1**	2	**Utotal**	0.89
S2	**C2**	1	**T2**	4		

S3	**C3**	1	**T3**	7		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

Analysis Explanation:

In example-6, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

Example-7 Analysis:

For **Example-7**, we have the following parameters:

Example-7						
S1	**C1**	1	**T1**	2	**Utotal**	1.00
S2	**C2**	1	**T2**	4		
S3	**C3**	2	**T3**	8		

For the above service/task information, the code proves that the scheduling is **FEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is **FEASIBLE**.

Analysis Explanation:

In example-7, the task set is FEASIBLE according to the code. The Cheddar analysis for all the scheduling policies (i.e. RM, EDF, and LLF) shows that the task set is schedulable. This is because no deadline is missed for any scheduling policy given the worst case execution times. The code results match with the respective Cheddar analysis. All service periods are suitable for the worst case service execution timings and RM scheduling thus is schedulable as all deadlines are met.

Example-8 Analysis:

For **Example-8**, we have the following parameters:

Example-8						
S1	**C1**	1	**T1**	3	**Utotal**	1.00
S2	**C2**	2	**T2**	6		
S3	**C3**	3	**T3**	9		

For the above service/task information, the code proves that the scheduling is **INFEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is not schedulable and therefore, **INFEASIBLE** as some service deadlines will be missed. However, the Cheddar Analysis for the EDF and LLF scheduling policies showcase that the service set is schedulable (as no deadlines were missed) and therefore, **FEASIBLE**.

Analysis Explanation:

In example-8, the task set is INFEASIBLE according to the code. The Cheddar RM analysis shows that some task deadlines will be missed and therefore the task set is NOT schedulable. This is true because the service S3 requires a capacity of 3 for a period of 9. Because of the RM scheduling policy of servicing high frequency tasks with higher priority, the service S3 with the lowest priority is not serviced before its deadline in its first cycle of a period of 9 units. The service S3 is serviced after its first deadline and therefore the task set is not schedulable. As opposed to the Cheddar RM analysis, the EDF and LLF Cheddar analysis shows that service priorities are set according to the amount of execution time left before the service deadlines are due (i.e. according to the EDF and LLF policies). Given that, the EDF and LLF Cheddar analysis allows the task set to meet ALL the deadlines and therefore the services are schedulable according to the EDF and LLF policies.

Example-9 Analysis:

For **Example-9**, we have the following parameters:

Example-9						
S1	**C1**	1	**T1**	4	**Utotal**	1.00
S2	**C2**	2	**T2**	8		
S3	**C3**	3	**T3**	12		
S4	**C4**	4	**T4**	16		

For the above service/task information, the code proves that the scheduling is **INFEASIBLE**. Cheddar analysis for the above information using the RM scheduling policy also proves that the service set is not schedulable and therefore, **INFEASIBLE** as some service deadlines will be missed. However, the Cheddar Analysis for the EDF and LLF scheduling policies showcase that the service set is schedulable (as no deadlines were missed) and therefore, **FEASIBLE**.

Analysis Explanation:

In example-9, the task set is INFEASIBLE according to the code. The Cheddar RM analysis shows that some task deadlines will be missed and therefore the task set is NOT schedulable. This is true because the service S4 requires a capacity of 4 for a period of 16. Because of the RM scheduling policy of servicing high frequency tasks with higher priority, the service S4 with the lowest priority is not serviced before its deadline in its first cycle of a period of 16 units. The service S4 is serviced after its first deadline and therefore the task set is not schedulable. As opposed to the Cheddar RM analysis, the EDF and LLF Cheddar analysis shows that service priorities are set according to the amount of execution time left before the service deadlines are due (i.e. according to the EDF and LLF policies). Given that, the EDF and LLF Cheddar analysis allows the task set to meet ALL the deadlines and therefore the services are schedulable according to the EDF and LLF policies.

SUMMARY:

From all the above explanation for examples, it is very clear that when the code N&S feasibility tests fail, the Cheddar RM analysis fails to provide a schedulable task set. That means, both the code output and the Cheddar RM analysis provides a similar viewpoint. On the contrary, for cases where the Cheddar RM analysis fails to provide a schedulable task set, the Cheddar analysis for EDF and LLF policies provide a schedulable output although the code N&S feasibility test calls the task set infeasible. This is because, the Cheddar EDF and LLF policies schedule the task set such that no deadline is missed given the worst case execution times for the task set and the assumptions as presented by Liu and Layland in their research work.

For the cases where the code N&S feasibility tests pass and call the task set as feasible, the Cheddar RM analysis provides a schedulable task set. That again means, both the code output and the Cheddar RM analysis provides a similar viewpoint. In addition to the Cheddar RM analysis, the Cheddar analysis for EDF and LLF policies also provide a schedulable output and that again is so because no deadline is missed given the worst case execution times for the task set and the assumptions as presented by Liu and Layland in their research work.

4) [15 points] Provide 3 constraints that are made on the RM LUB derivation and 3 assumptions as documented in the Liu and Layland paper and in Chapter 3 of RTECS with Linux and RTOS. Finally, list 3 key derivation steps in the RM LUB derivation that you either do not understand or that you would consider "tricky" math. Attempt to describe the rationale for those steps as best you can do based upon reading in Chapter 3 of RTECS with Linux and RTOS.

Sol.

#3 Constraints, #3 Assumptions

The #3 constraints and #3 assumptions that are made on the RM LUB derivation as in the book and in the paper are as follows:

#1 Constraint, #1 Assumption:

Assumption: Each task with a hard-deadline will be periodic with a constant interval between its requests and each such task must be completed before the system receives the next request for that task.

Constraint: The RM LUB theory does establish an assumption regarding the periodicity of the service requests and that these services will be requested at constant intervals. However, the RM LUB theory does not account for the differential timing that is often non-deterministic which the services require to begin its execution before the service execution begins and the service execution ends. The issues of IO coupling are not considered and therefore the RM LUB theory is best suited to IO decoupled services that have limited buffering non-determinism and latencies associated with the task/service calls and switching processes. Also, the RM LUB theory does not support the fact that in hard RT-systems, responses of the system are critical and therefore to prevent a complete system shutdown, if one deadline has been missed, the scheduler must drop that service and move on to its second rest to prevent other services from missing deadlines.

#2 Constraint, #2 Assumption:

Assumption: The execution time (run-time) for each task remains constant irrespective of time. The task period is assumed equivalent to the task deadline.

Constraint: The RM LUB theory supports the assumption that execution times of services remains constant irrespective of time but as mentioned in the previous constraint, it does not account for the system buffering latencies and IO coupling non-deterministic responses. Often for RT-systems, IO decoupling is performed to reduce the uncertainty in the scheduling. However, the use of synchronization primitives and/or preventive methods (like semaphores, mutexes, etc.) often add complexities to the RM LUB theory which it may not account for. Therefore, the execution time for each task can be deterministic but may not be always constant irrespective of time. The other assumption made where the task period is equal to the task deadline does not consider the latencies associated with the task deadline which is not a part of task period. Since, the tasks are serviced periodically with the task period considered equal to the task deadline, meeting the RM LUB theory would is pessimistic as it involves the worst case execution time pertaining to the task deadline. This is where the DM (Deadline Monotonic) theory differs from that of RM.

#3 Constraint, #3 Assumption:

Assumption: Each task is independent of the initiation or completion requests of other tasks.

Constraint: The RM LUB theory assumes that each task is independent of other task. However, in complicated hard RT-systems, concurrency programs often involve tasks and service sets that are non-independent and that their execution relies on each other through the use of protected critical resources. Such concurrent environments may not be suitable for the assumption made in the RM LUB theory, the reason being that a task may wait for another task for a shared resource while the other task may or may not be a real-time task and that it may have its own different dependencies that the RM LUB theory may not account for.

#3 Key Tricky Steps

The three key derivation steps in the RM LUB derivation that can be considered as tricky is as under:

1. The step leading to the derivation of the simple expression for utility based on the fractional interference is a tricky one. The equation is as under which can be found in the RTECS book:

$$U = 1 - \left(\frac{f(1-f)}{(T_2 / T_1)} \right)$$

2. The equation giving the difference of the utilization factors as a function of the service periods is a tricky one to understand. The equation is as under which can be found in the Liu and Layland paper:

$$U - U'' = -(\Delta/T_1) + (2\Delta/T_2) > 0.$$

3. The differential equation where the first derivative of the Utilization factor U is take with respect to the g's is complicated and tricky to catch. The equation is as under which can be found in the Liu and Layland paper:

$$\partial U/\partial g_j = (g_j^2 + 2g_j - g_{j-1})/(g_j + 1)^2 - (g_{j+1})/(g_{j+1} + 1) = 0, \quad j = 1, 2, \cdots, m - 1.$$

Challenges with Excerise-2:

- Setting up the Cheddar open-source real-time scheduler tool required some practice and time.
- Understanding the feasibility example code and the functions in the code required decent time and effort. The functions that were of importance are as under:
  ```
  int completion_time_feasibility(U32_T numServices, U32_T period[], U32_T
  wcet[], U32_T deadline[])
  ```
  and
  ```
  int scheduling_point_feasibility(U32_T numServices, U32_T period[], U32_T
  wcet[], U32_T deadline[])
  ```
- In addition to the above challenges, understanding the cheddar analysis and the code output also required a decent amount of effort.
- Understanding and summarizing the PASS research paper took fairly a large amount of time.
- It has been difficult to understand some of the equations and derivations in the Liu and Layland research paper. I have my best efforts to provide the three key steps that I still consider as tricky.