ECEE 5623, Real-Time Systems:
Exercise #2 – Service Scheduling Feasibility

Michelle Joslin Christian

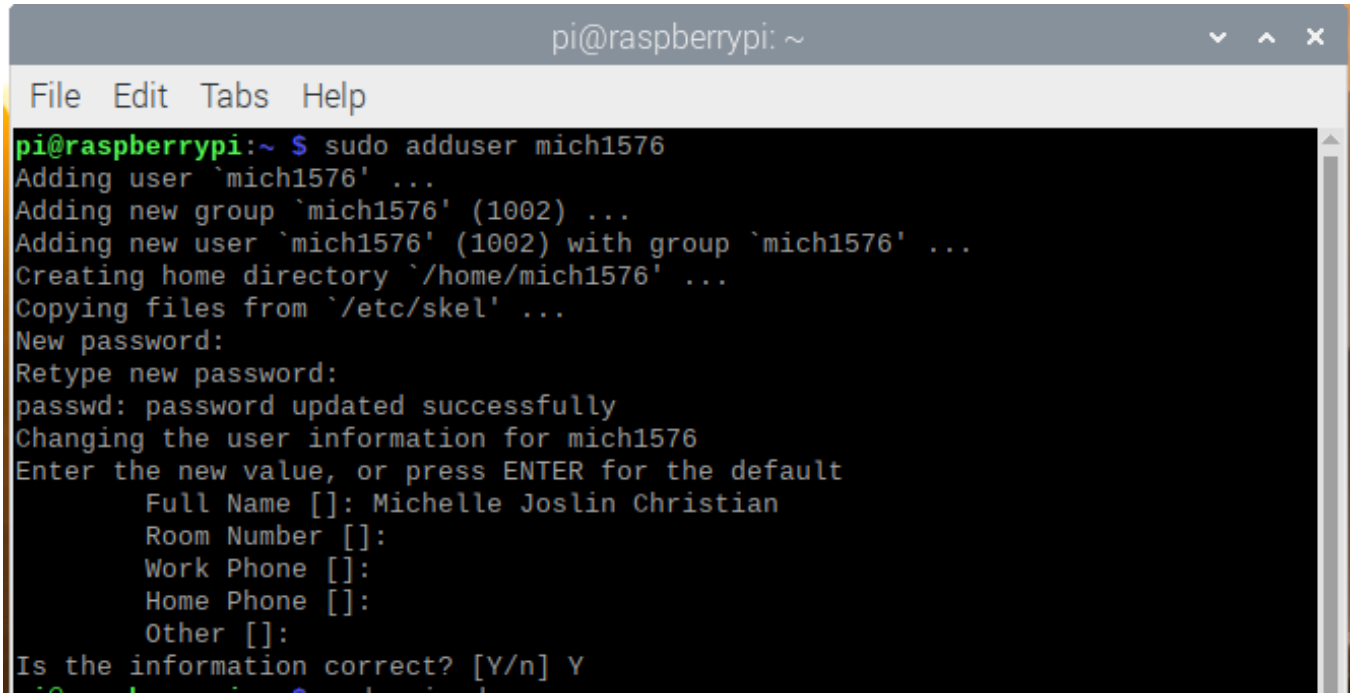
Github URL: https://github.com/Mich2899/RTES_exercise2

Question 1) If you're using embedded Linux, make yourself an account on your R-Pi3b.
a) Account creation - To do this 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", "VS code" or "geany" for Unix systems (you can normally do "sudo apt-get install geany" or any alternative editor you like best). You are welcome to use whatever editor works best for you in this class. Do a quick "sudo whoami" to demonstrate success for account creation.

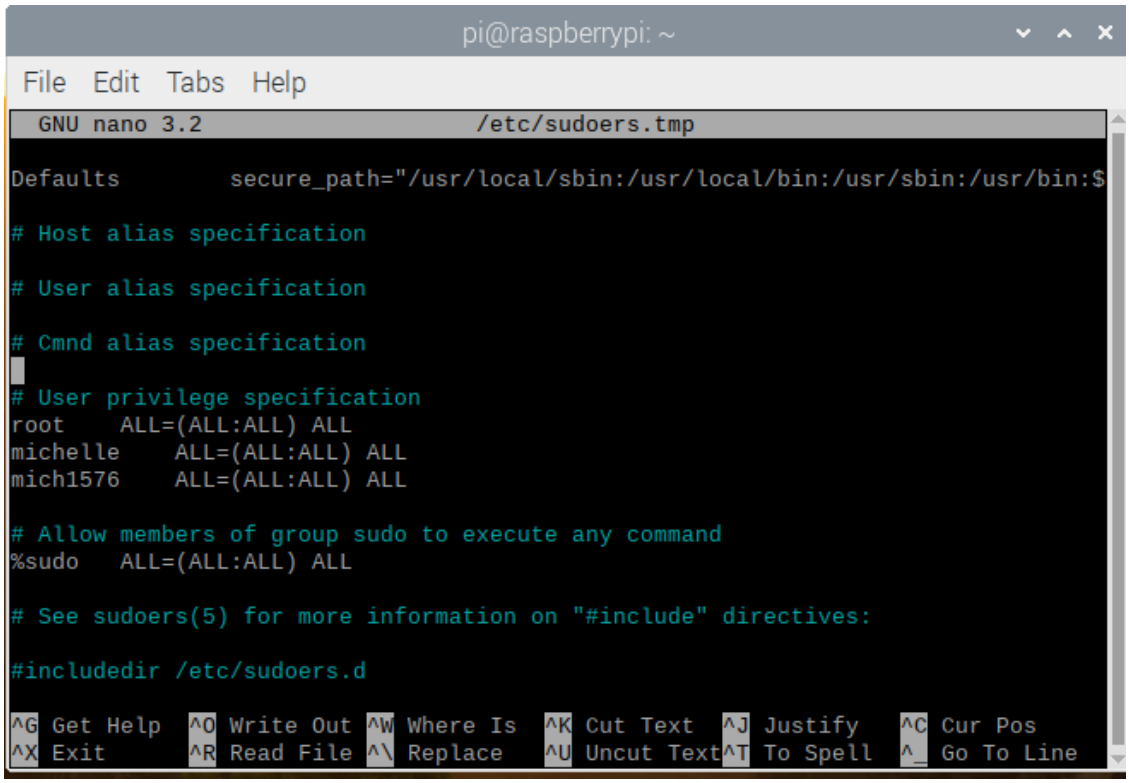
Steps for adding a user into raspberry pi:

1.) **sudo adduser <*username*>**

- sudo adduser allows to add users into raspberry pi apart from the default user (usually pi).
- As we run this command into the terminal it adds a user and creates a home directory besides the deafult one and demands for a new password for this user login.
- Some other details to be entered are full name of the user, room number, work phone, home phone and any other data user wishes to enter. Once can leave all these categories/fields blank and still create an account. At the end of all the fields, a message pops up asking if the data entered is correct or not.

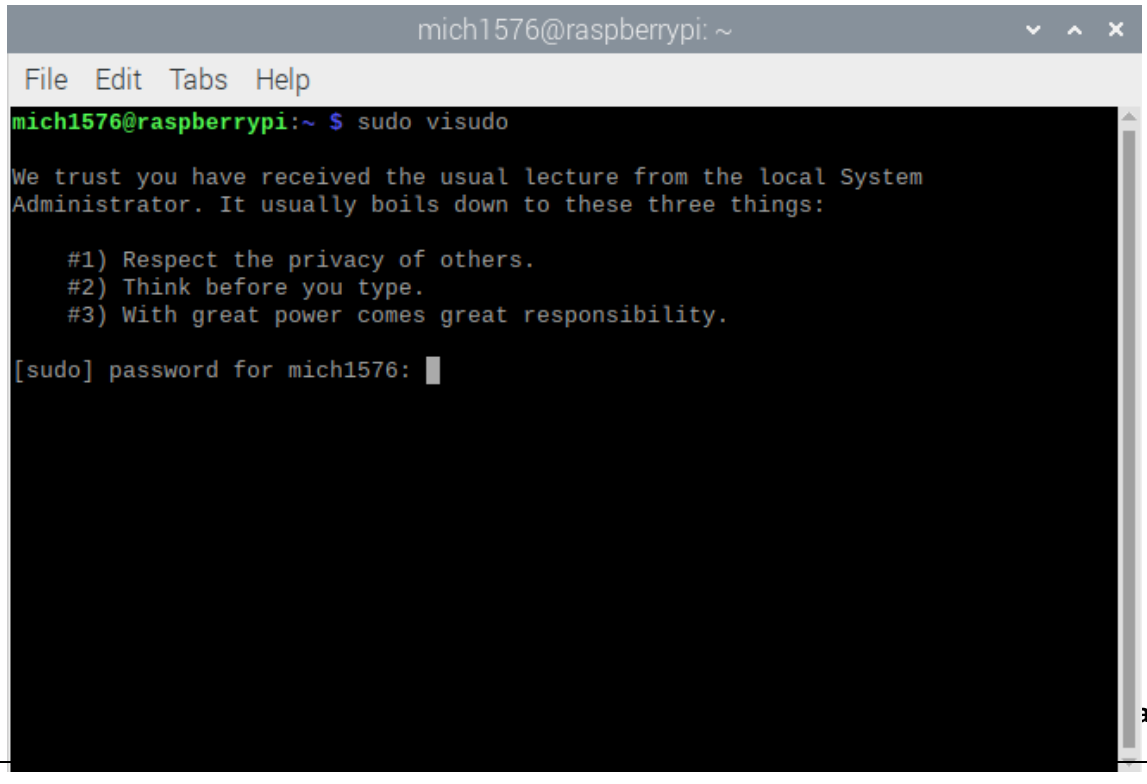


- To provide the same privileges to the newly added account we run the “sudo visudo” command and add the username with same privileges as root.



root *ALL= (ALL:ALL) ALL*
<username> *ALL= (ALL:ALL) ALL*

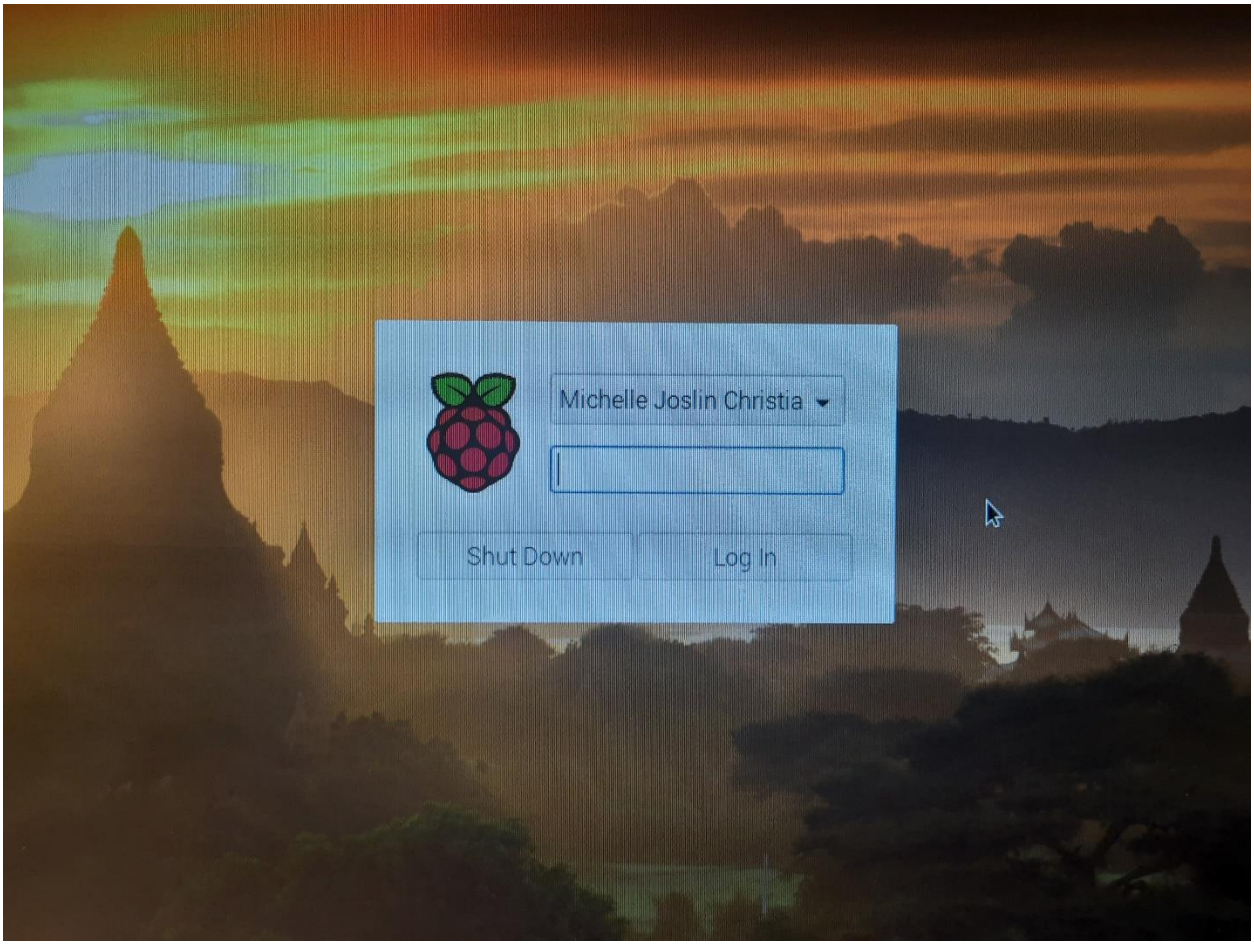
- As shown in the above screenshot I have added two accounts to Raspberry pi with usernames, “michelle” and “mich1576” and provided the same privileges as the root user.



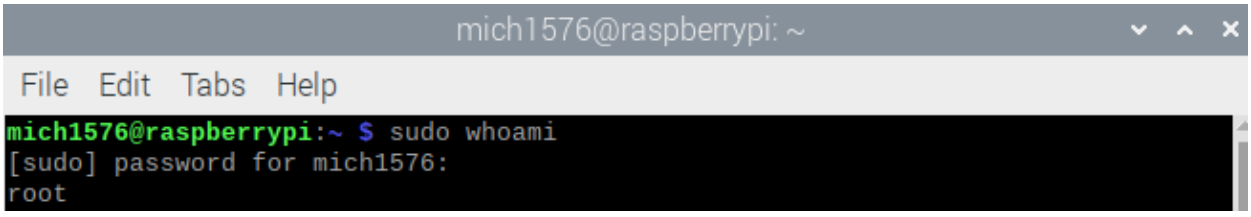
- This screenshot is an affirmation that the new account is created and is secured by a password.

b) Logout 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. Show evidence that you have done this with screenshots or photos with your smart phone.

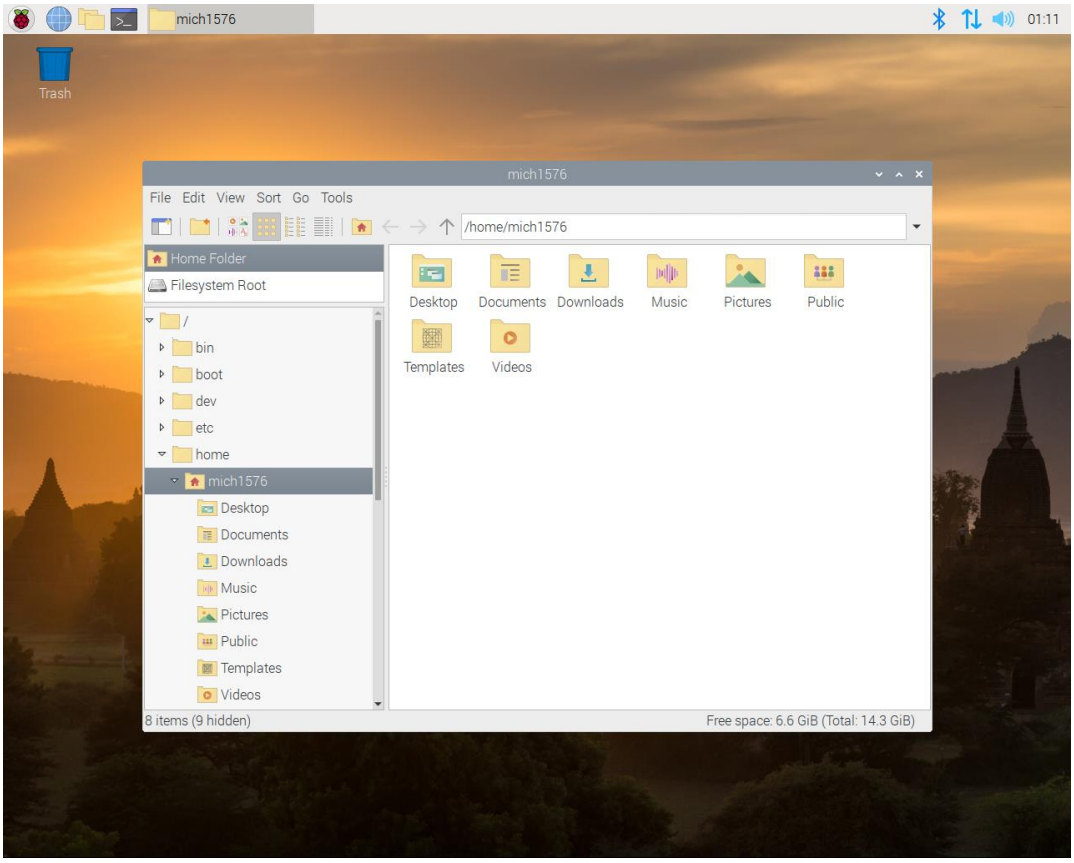
- After logging out of the pi account the current screen appears where it provides all the available accounts and to access them we need the respective passwords.



-Running the “sudo whoami” command on the terminal provides this output.



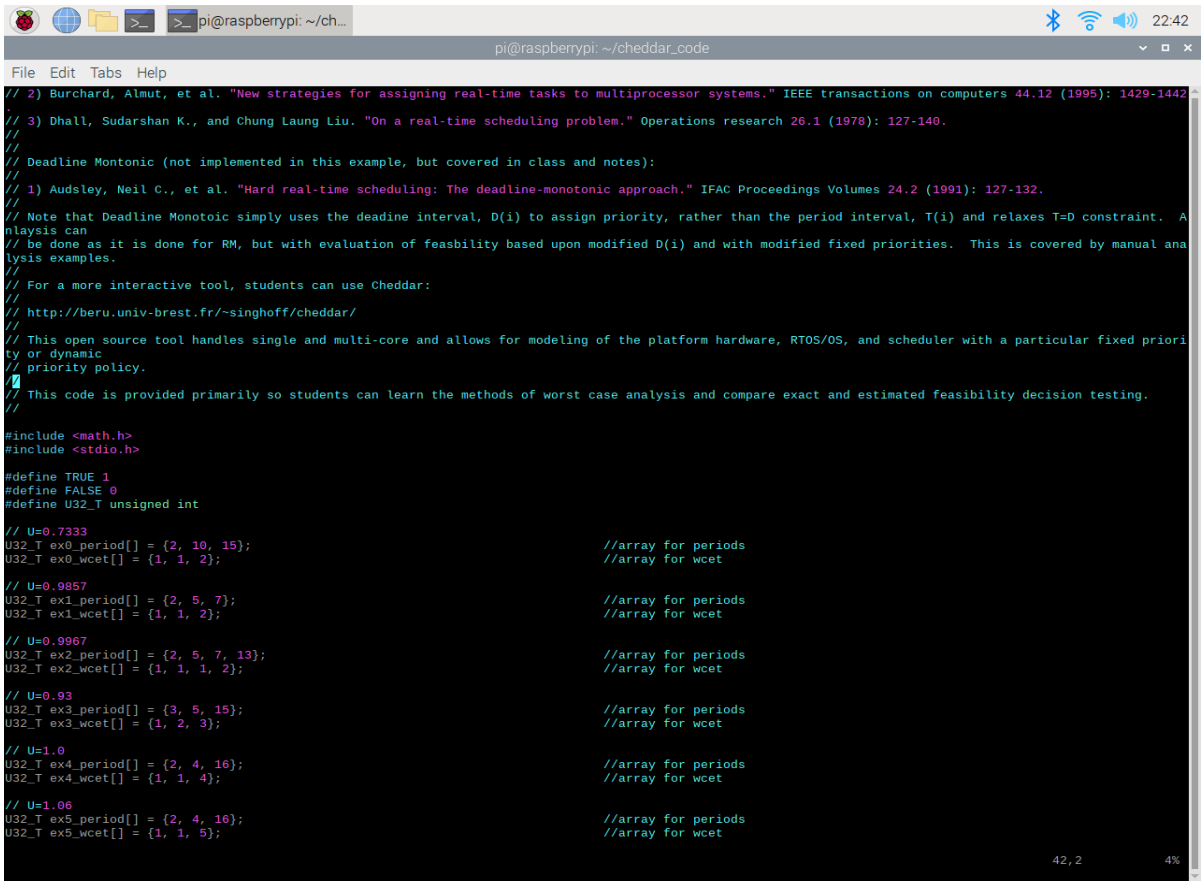
- After new account if created, raspi creates a home directory for the new user as shown below.



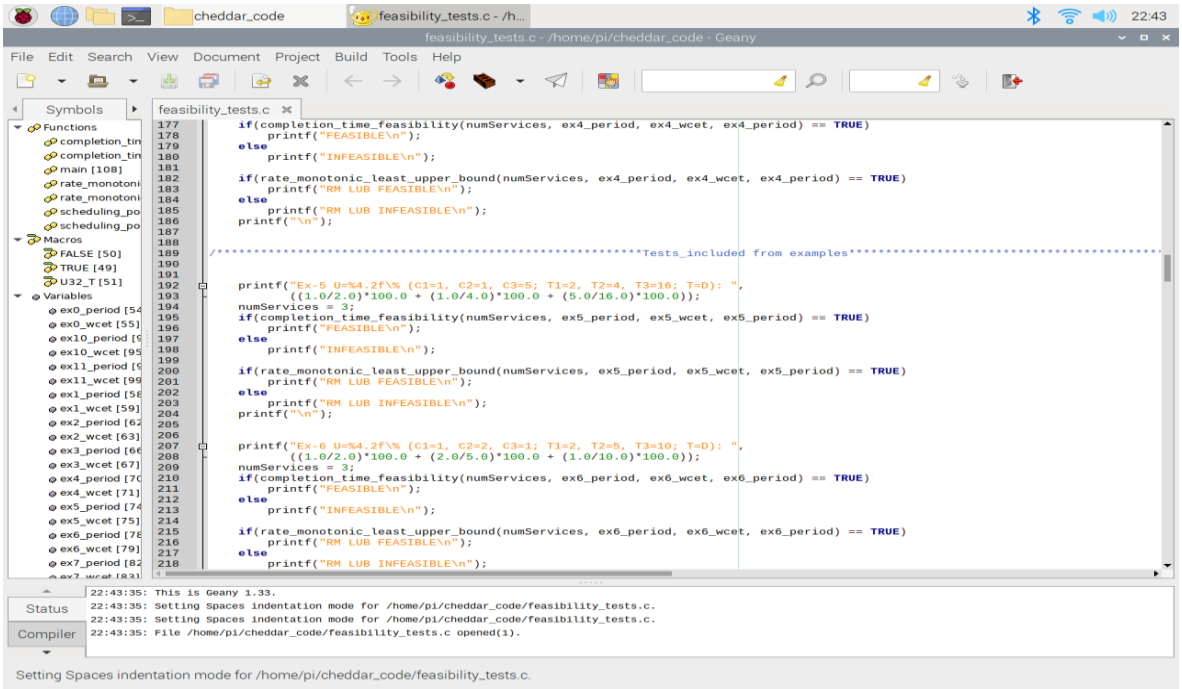
- I have a lab setup (monitor, keyboard and mouse) just for raspberry pi and hence I do not require remote access but I understand and know the use of MobaxTerm and accessing Raspberry Pi desktop using SSH and VNC.

c) 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.

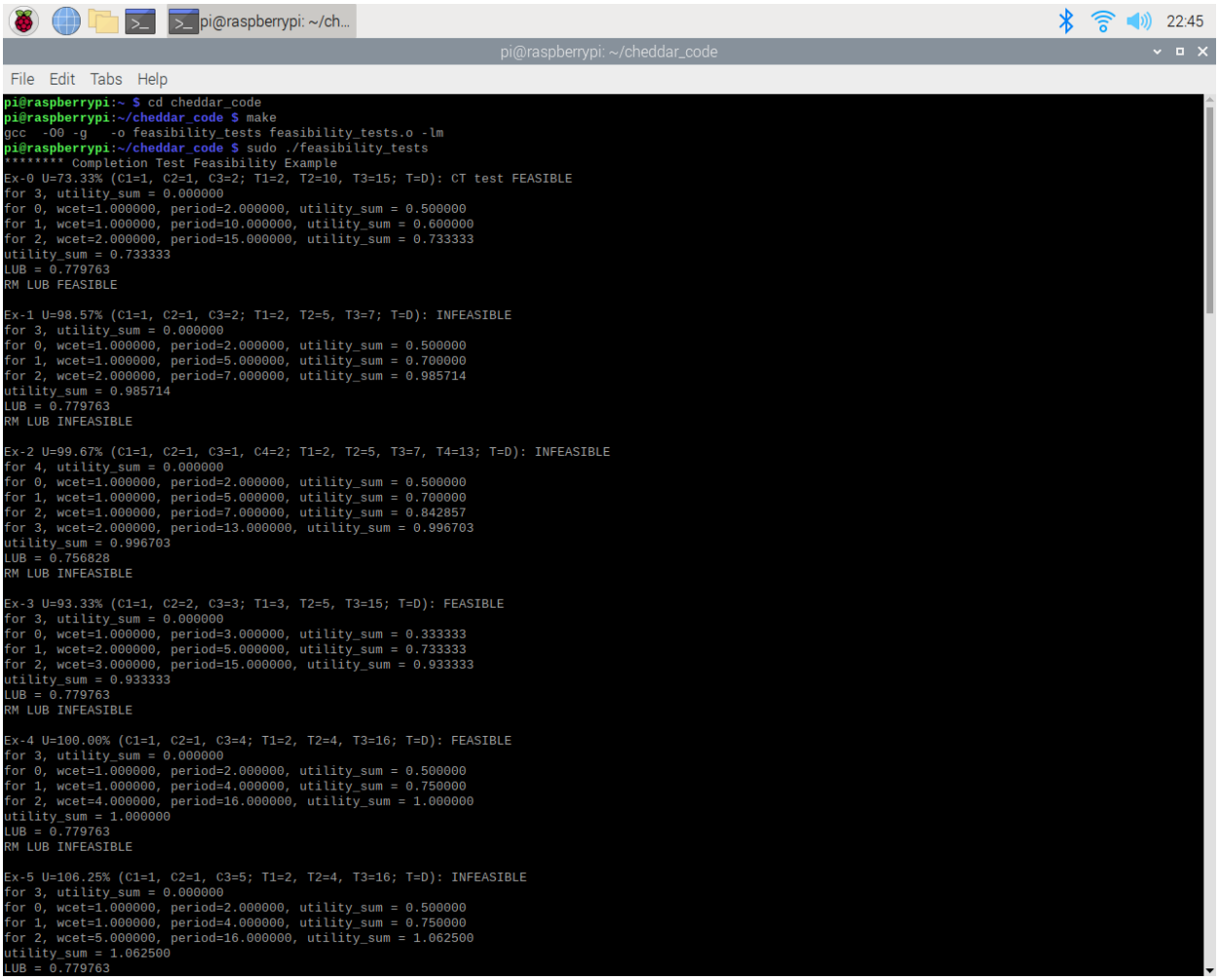
- The default browser on my raspberry pi (3B+) is *duckduckgo.*where I have set the default page and the home page to the course website.



- I usually use vi editor or geany for editing my codes.



- compile/make and run the file on the terminal.



Question 2) Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" , by Gene Carlow.

a) Provide an explanation and critique of the frequency executive architecture.

The paper "*Architecture of the Space Shuttle Primary Avionics Software System*", discusses about the PASS software architecture and the programming techniques used for building the software system. PASS software program is said to be one of the most complex fight computer programs ever developed. It employs computer redundancy management concepts along with a numerous amount of functions, each dedicated to perform a specific functionality. It implements a relatively simple streamlined operating system or executive, but it is very critical to take in account the synchronization of start and stop for different application processes. The associated I/O must be taken care of as well to avoid overruns at both process and system level. The executive used is simple and hence very deterministic. The repeatability of code is the major benefit of this approach. Although, for such a simple yet complex (a multitude of functionalities to be implemented), the flexibility decreases to a great extent. To accommodate any change would be difficult for such a routine. The number and variety of different applications that Shuttle computers are required to perform may increase. Along with this, there are also expectations that the requirements would change at any point, if the program has to be evolved. The PASS architectural structure reflects a blend of operational, reliability and functional requirements along with several physical constraints. To meet the resource and reliability constraints, the architecture adopts and operational structure that is based on the combination of mission phases and major application functional requirements. These segmentations divide PASS into eight different phase/ function combinations, each of which is identifies with a unique operational sequence (OPS) designation. These operational sequences are further divided into different modes to implement different level of functionalities based on their importance and requirement in the program. For e.g. the primary processing is handled by major modes while element like SPEC (specialist function) handle the secondary or background functions. Similarly, the display function is managed by DISP substructure of OPS. These substructures are further divided into substructures to provide functionalities like man/machine interface and more.

The flight systems software architecture reflects a synchronous design approach within which, the dispatching of each application process is timed to always occur at a specific point relative to the start of an overall system cycle or loop. This architecture can also be defined as Cyclic executive or the Frequency executive. Cyclic executive is a control structure that explicitly interleaves the execution of more than one periodic process on a single CPU. This kind of executive is usually implemented using a main loop with an invariant body known as the cyclic schedule. The major drawback of this architecture is that it does not support asynchronous interrupts and hence may be useful for simple applications but for complex structures such an architecture may not be useful. To accommodate asynchronous interrupts into the existing skeleton of this executive, it can include ISRs (Interrupt service routine) to select one of the several loop invariants based upon event data, often called as Main+ISR. In Main+ISR several loop invariant bodies selected by input data can provide different modes or rates of function execution for example, high medium and low frequency executive.

Frequency executive architecture on the PASS system can be better explained using function implementations. For example, functions like process management in the FCOS (Flight Computer operating system), control the allocation of computer resources. It manages the resource requests from other systems or application processes. These requests are made through a standard set of

service interfaces (SVCs) with reference to specifications that define the time and or/event, frequency, and priorities for servicing the request. Process phasing and overall systems load balancing are maintained through the management of specifications. The processing management function ensures that the highest priority system or application process, ready with work to perform is given the CPU resources required to accomplish it. Different processors are used to manage different operations like IOP (input/output processor) to handle the IO functions.

The system uses multiple computers, each of them having a simple implementation of the respective functionalities. The major effort/ problem is synchronizing these in a redundant configuration. To establish a communication between these sections, interfaces are implemented. A system-level cyclic process is initiated to continuously service these interfaces and establish the timing cycle that is inherent in the avionic system design. The magnum opus would be developing the flow control to have a more efficient transition between one process to another or one OPS to another. This minimizes the flexibility of the system. Adding any new feature interrupts the whole program flow. This increases the chances for a service to be delayed and hence increases the chances of missing the deadline.

The control segment decides the overall sequencing and control for each application. It defines the initiation, sequencing and termination for each operational sequence, major mode, and SPEC. One more example of the closed loop executive present on the PASS system is the GNC application. It is a cyclic closed loop application, which performs variety of functions with extremely tight timing and phasing relationships. The execution rates for the principle functions within an OPS vary from 25 z for support of the basic vehicle flight control, down to 0.25 Hz for display update. Executive cyclic processing is initiated by the OPS control segment using an FCOS process scheduling SVC. The executive uses a table-driven dispatching design technique to initiate and control the processing of principal functions.

The high frequency executive is schedules at a relatively high priority to cycle at a 25 Hz rate and initiate all principal function processes directly related to vehicle flight control. Mid-frequency and low-frequency executives are scheduled at low priorities. They initiate principal function processes, which operate at rates of 6.25Hz down to 0.25 Hz. The dispatcher in the system management application for special processing operates at a rate of 5Hz.

In conclusion, the frequency executive architecture provides a simple approach to implementing applications. As it has applications running in a periodic loop operated at specific instants of time, it makes the system simple and deterministic and synchronous. It is easier to determine what is expect for such a repetitive algorithm. Although the simplicity of applications includes the complexity of interfacing multiple of these for the desired result. It is a continuous loop of instructions and hence it cannot be preempted for a higher priority task. There is no context switch for such executives and hence it saves a lot of overhead.

b) 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.

The frequency or the cyclic executive (also known as clock driven), executes an application which is divided into a sequence of non-preemptible tasks, involving each task in a fixed order throughout the execution history of the program. It repeats the task list at a specific rate (frequency), in which the subtasks or the minor tasks each have their own unique rate of execution. The period of every task in the main cycle is assumed to be periodic to simplify the implementation, which can be triggered by a periodic timer or by some other periodic external hardware interrupt which keeps the program in synchronization with the external environment. Hence, as the name "cyclic" itself suggests, the flow of the program runs through control loop cycles through a predetermined path in a very predictable manner. Although, this type of methodology fails to execute when there are asynchronous interrupts or unpredictable changes in the task frequency. Thus, the architecture has its own advantages and disadvantages as listed below:

Advantages:

1. **Deterministic**
   It is very clear from the definition that the entire flow of the application is predetermined, assuming there is no overload. Thus, this means that it is possible to predict the entire future history of the state of the machine, once the start time of the system is determined. Thus, if the response requirements of all the tasks are met without missing any deadlines, the framework proves to be quite efficient.

2. **No preemption required/ Minimum or no overhead**
   There is no interrupt request from another tasks while execution of an individual task, there is no requirement for application task preemption. Thus, executive overhead can be kept low because there are no unexpected context switches. There is no requirement for allowing external interrupts since no tasks will process the event until its cycle begins and the polling mechanism generates the interrupt synchronously. Along with this, as along as we do not use shared memory multiprocessor, we do not need to implement method to protect the shared data structures (e.g. Semaphores, rendezvous, monitors). These benefits into a software which is very simple, efficient and generally fast.

3. **Low jitter**
   The determinism and simplicity of the executive provides the advantage of low jitter. Jitter is the variation in output delay for each iteration (delay is not constant). As there is low-jitter since the start of the program, the results execute a highly precise time relationship relative to is previous completion time and its succeeding completion time, such that there will be a suitably small change in the time variation from frame to frame (drift will also be low).

Disadvantages:

1. **Flexibility**
   One of the major drawbacks of cyclic/ frequency executives are that these architectures are very fragile. Accommodating any change into such a system during the integration or later during the maintenance of the system when additional functionality has to be implemented results into some task missing its deadline. It is very difficult and error prone to make any amendments to such architecture.

2. **Frequency**
   Another downside of cyclic/frequency executive is implementing task with different frequency and resource requirements. For example, one task requires significant processing time to extract the correct information while the other has a higher request rate. For such applications the task with longer processing time is divided into subtasks to fit between the request period of higher frequency task.

3. **Effective only for simple systems**
   For applications that include asynchronous interrupts and aperiodic set of tasks, cyclic/frequency executive turns out to be an inefficient option. It is very difficult to develop systems of any complexity with this approach as this model does not support arbitray task processing rates. Cyclic executives require simpler control flow relationships to be hard coded into the program flow from task with lower frequencies to the one with higher frequency.

Question 3) Download feasibility example code and build it on your Raspberry Pi or alternate system of your choice and execute the code.
a) Compare the tests provided to analysis using Cheddar for the first 4 examples tested by the code.

The set of tasks taken into consideration
SET 0:

S1	T1= 2	C1=1
S2	T2=10	C2=1
S3	T3=15	C3=2

RM Scheduling: **FEASIBLE**
EDF Scheduling: **FEASIBLE**
LLF Scheduling: **FEASIBLE**

As observed in the timing diagram and the code test results below, the given set of tasks is schedulable using Rate monotonic policy with the utilization factor below the RMLUB and hence passes the RMLUB tests ass well (leaving slack time for the CPU). The set of tasks also pass the completion and the scheduling point feasibility tests. The timing diagram for RM given in the excel sheet matches the cheddar results.

Example 0	Service	Period	10 msec	Freq f	f0 multiple	Priority	WCET		Utility							
	S1	T1	2	0.5	7.5	1	C1	1	U1	50.00%	LCM =	30				
	S2	T2	10	0.1	1.5	2	C2	1	U2	10.00%	LUB =	77.98%				
	S3	T3	15	0.066667	1	3	C3	2	U3	13.33%	Utot =	73.33%	Slack	26.67%	Slack+U	100.00%

RM Schedule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S1								SLACK		SLACK				SLACK		
S2																
S3																

RASPI OUTPUT:

Completion Test

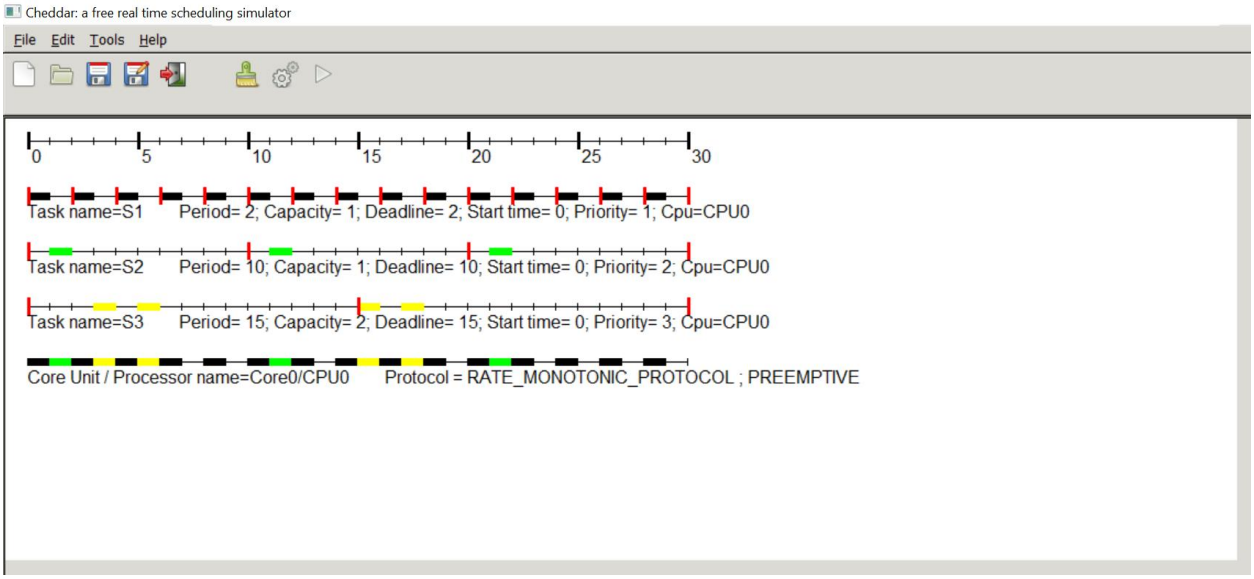
```
pi@raspberrypi:~/cheddar_code $ sudo ./feasibility_tests
******** Completion Test Feasibility Example
Ex-0 U=73.33% (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): CT test FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=10.000000, utility_sum = 0.600000
for 2, wcet=2.000000, period=15.000000, utility_sum = 0.733333
utility_sum = 0.733333
LUB = 0.779763
RM LUB FEASIBLE
```

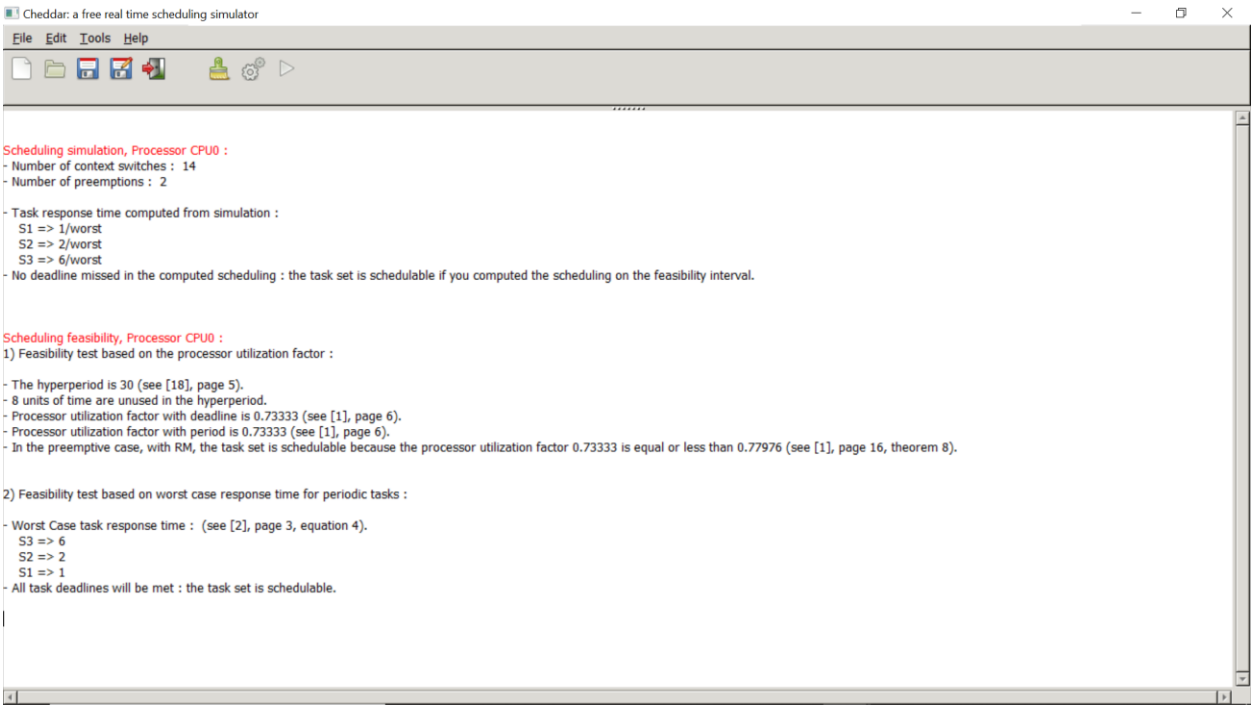
Sufficient Test

```
******** Scheduling Point Feasibility Example
Ex-0 U=73.33% (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=10.000000, utility_sum = 0.600000
for 2, wcet=2.000000, period=15.000000, utility_sum = 0.733333
utility_sum = 0.733333
LUB = 0.779763
RM LUB FEASIBLE
```

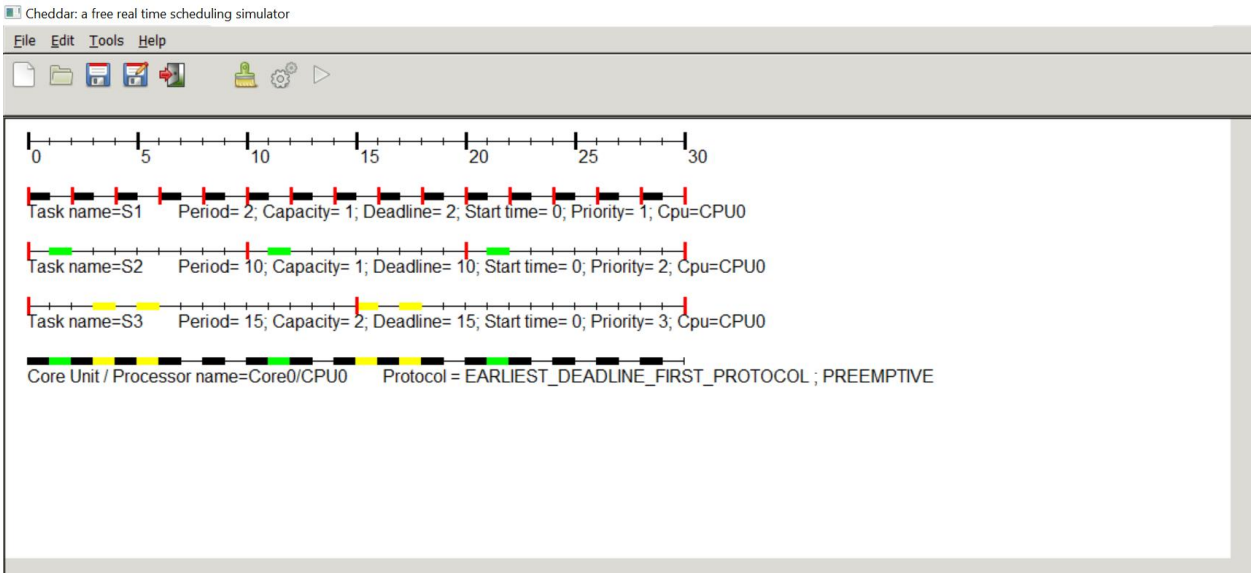
Cheddar results

Example 0 RM (Matches the timing diagram provided in excel sheet and the test results from the code)

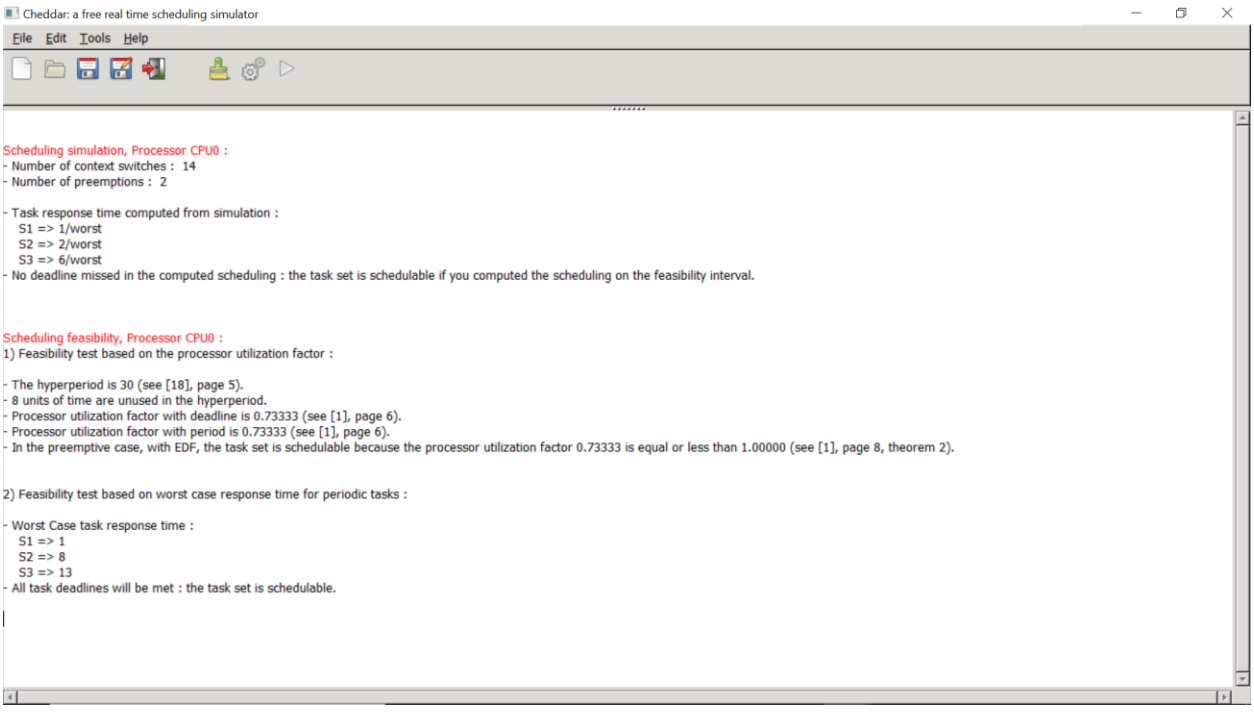




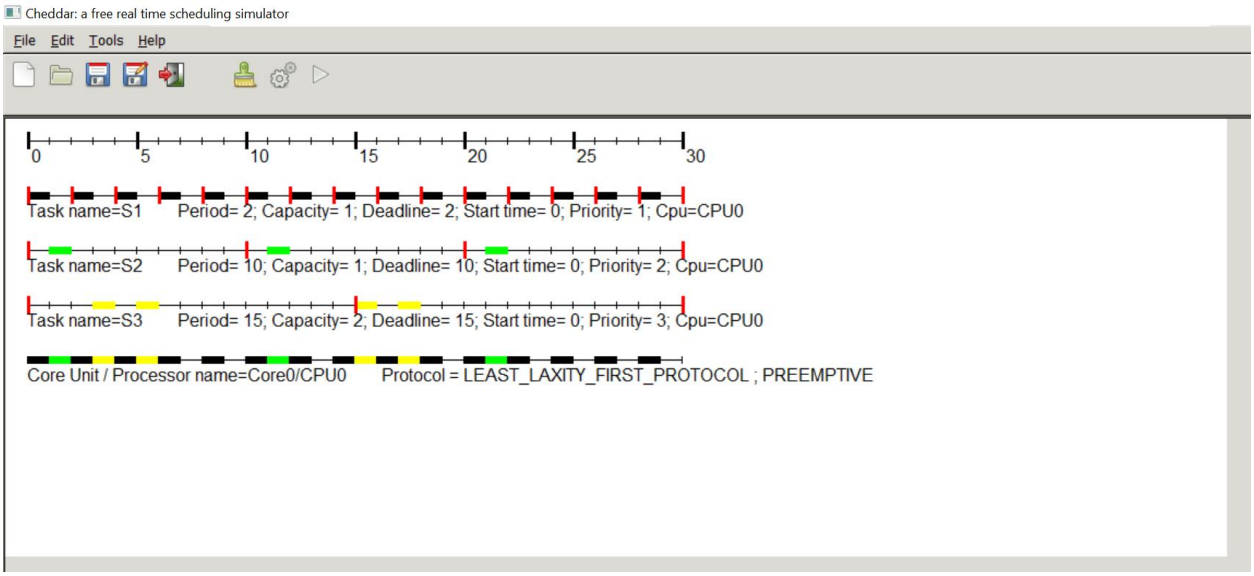
Example 0 EDF



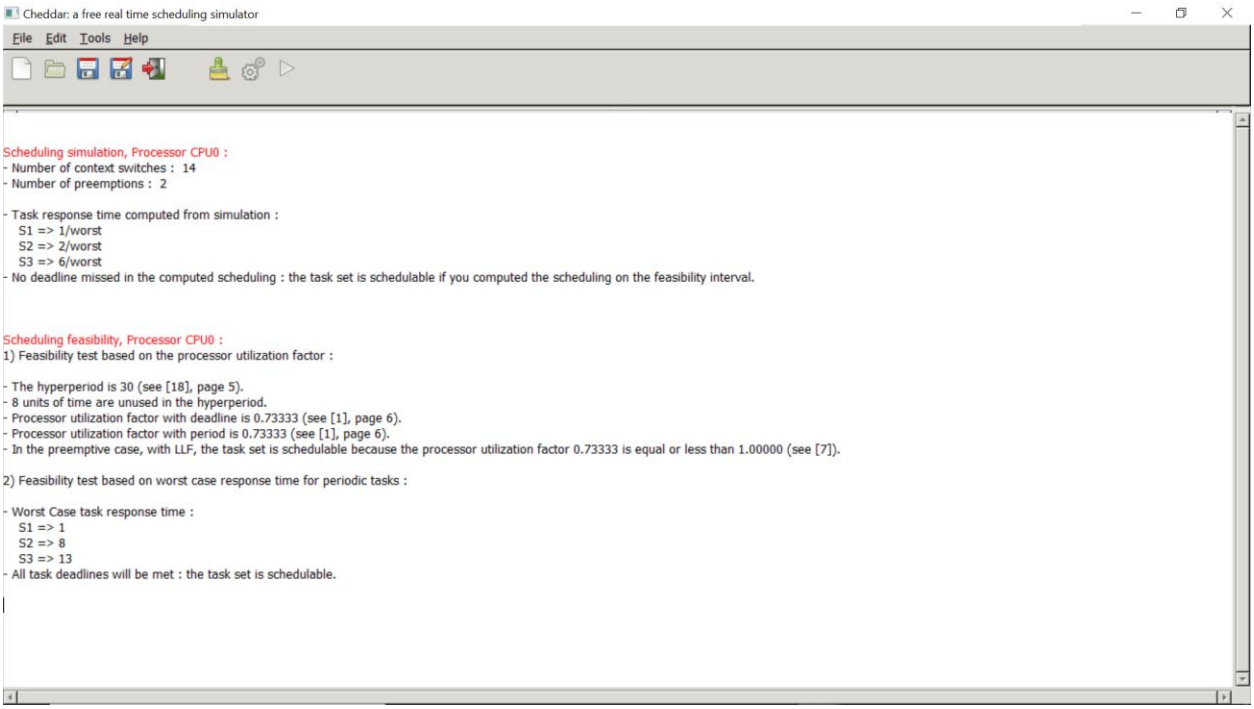
As shown in the above timing diagram and the test results for scheduling simulation and scheduling feasibility tests from Cheddar, given set of tasks is schedulable using EDF policy.



Example 0 LLF



As shown in the above timing diagram and the test results for scheduling simulation and scheduling feasibility tests from Cheddar, given set of tasks is schedulable using LLF policy.



The set of tasks taken into consideration

SET 1:

S1	T1= 2	C1=1
S2	T2=5	C2=1
S3	T3=7	C3=2

RM Scheduling: INFEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

The code output for the given set of tasks gives “infeasible” results for both completion tests and scheduling point feasibility tests. The utilization factor is 0.9857 (98.57%) which is much higher than the ideal RMLUB value for a set of three tasks which is 77.9763. Hence it fails the RMLUB tests. However, it is possible to schedule these tasks using the EDF and LLF scheduling policies. The cheddar output and the timing diagram for all three scheduling policies are provided below:

RASPI OUTPUT:

Completion test

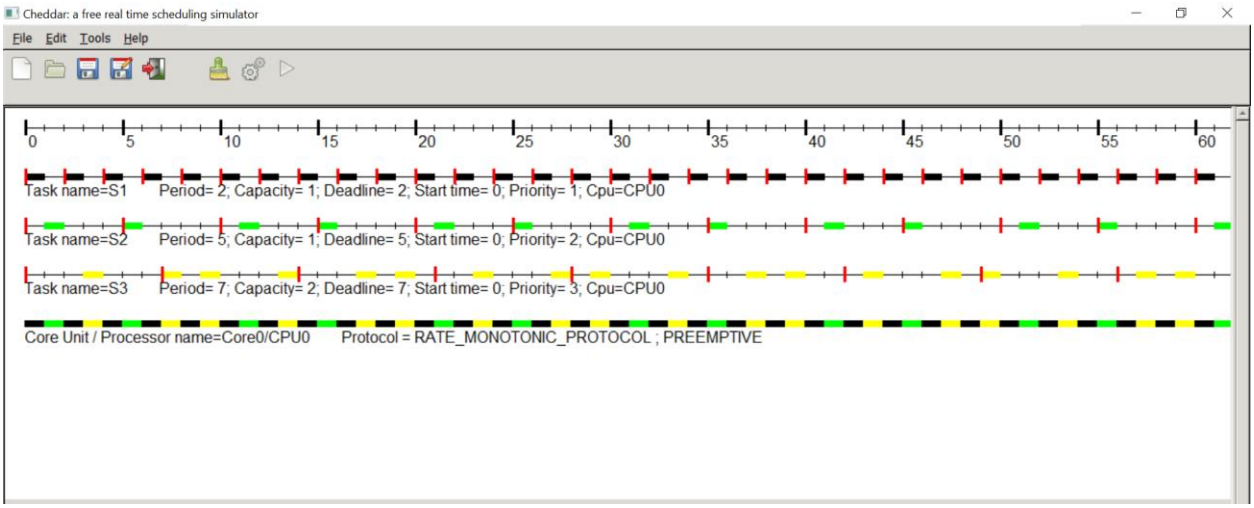
```
Ex-1 U=98.57% (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=2.000000, period=7.000000, utility_sum = 0.985714
utility_sum = 0.985714
LUB = 0.779763
RM LUB INFEASIBLE
```

Scheduling Test

```
Ex-1 U=98.57% (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=2.000000, period=7.000000, utility_sum = 0.985714
utility_sum = 0.985714
LUB = 0.779763
RM LUB INFEASIBLE
```

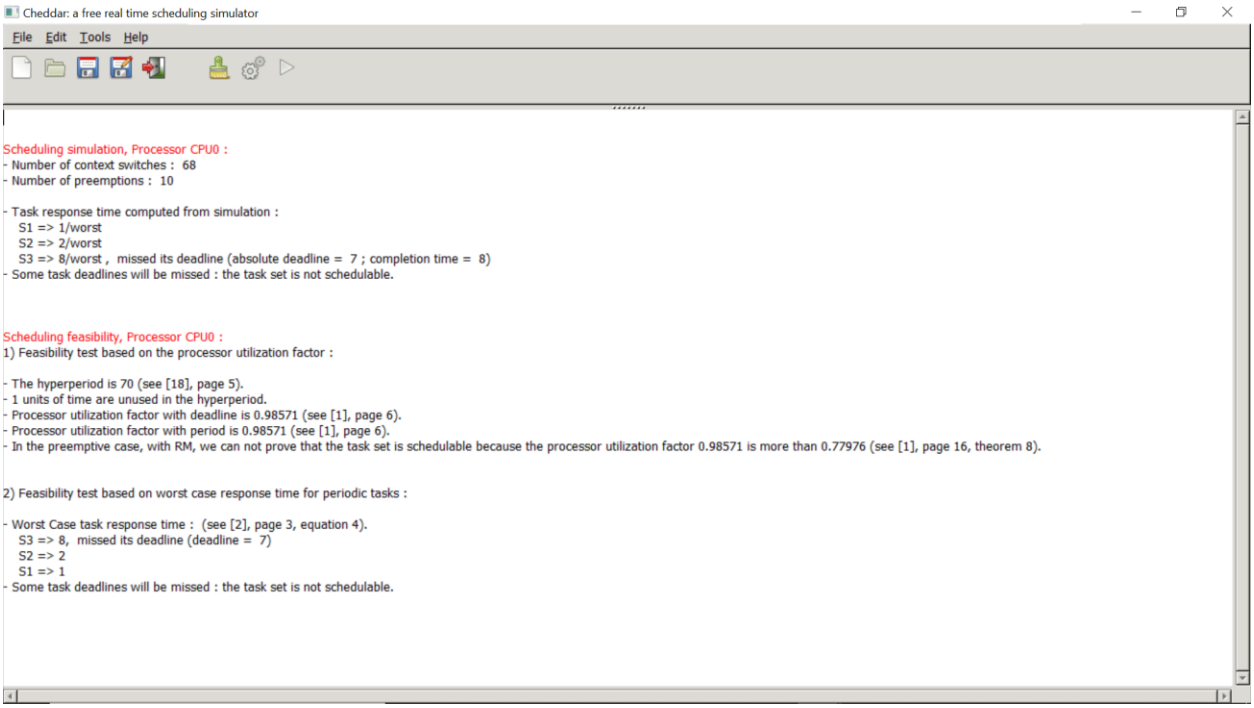
Cheddar results

Example 1 RM: The policy misses the deadline for the third service in the given request period and hence fails to schedule the set of tasks.

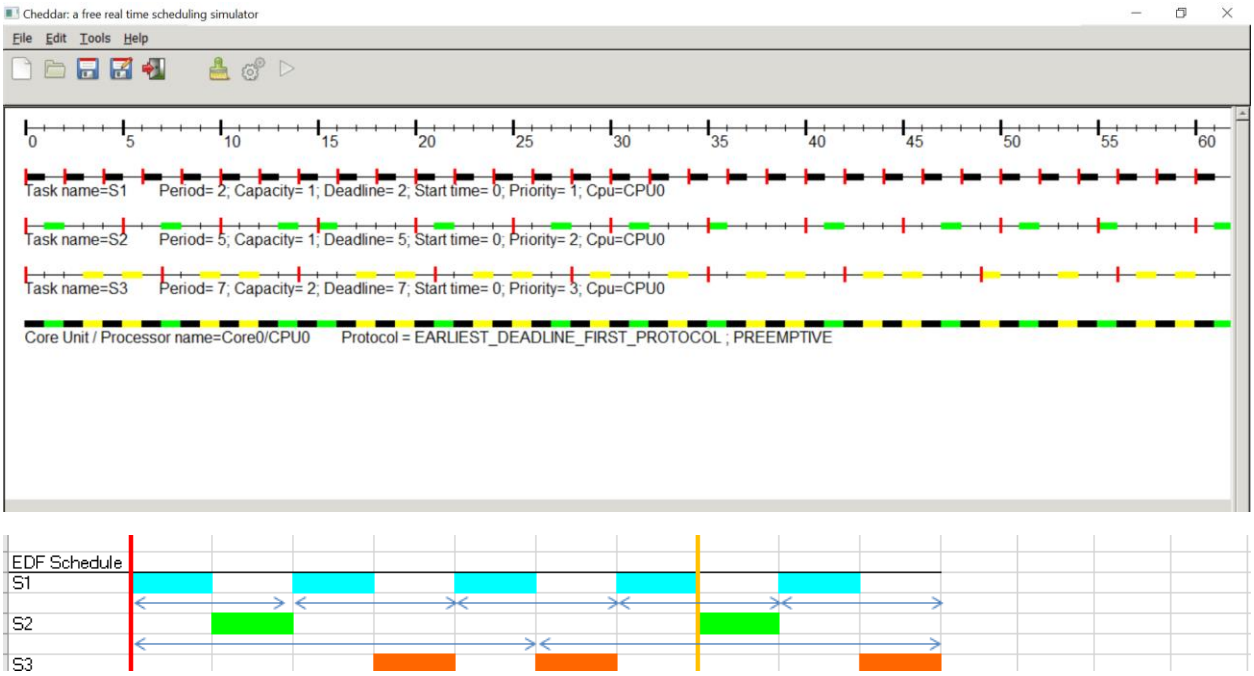


Example 1	Service	Period		Freq f	f0 multiple	WCET		Utility			
	S1	T1	2	0.5	3.5	C1	1	U1	50.00%	LCM =	70
	S2	T2	5	0.2	1.4	C2	1	U2	20.00%	LUB =	77.98%
	S3	T3	7	0.142857	1	C3	2	U3	28.57%	Utot =	98.57%

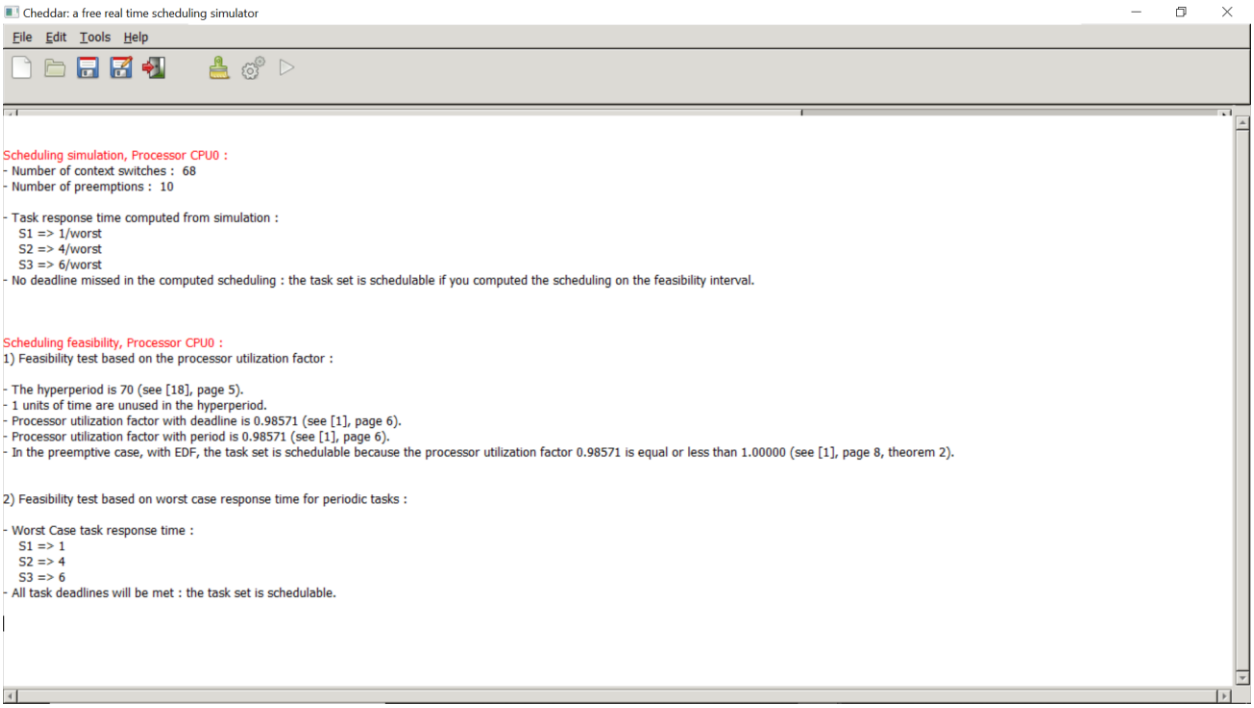
RM Schedule	1	2	3	4	5	6	7	8	9	10
S1							????????			
S2						????????				
S3								LATE		



Example 1 EDF: The timing diagram given in the excel sheet matches the cheddar results.



As shown in the scheduling simulation and the scheduling feasibility results provided by cheddar tool, the task set is schedulable using EDF if computed in the feasibility interval. The worst case response times for services are 1, 4 and 6 for service 1, 2 and 3 respectively.

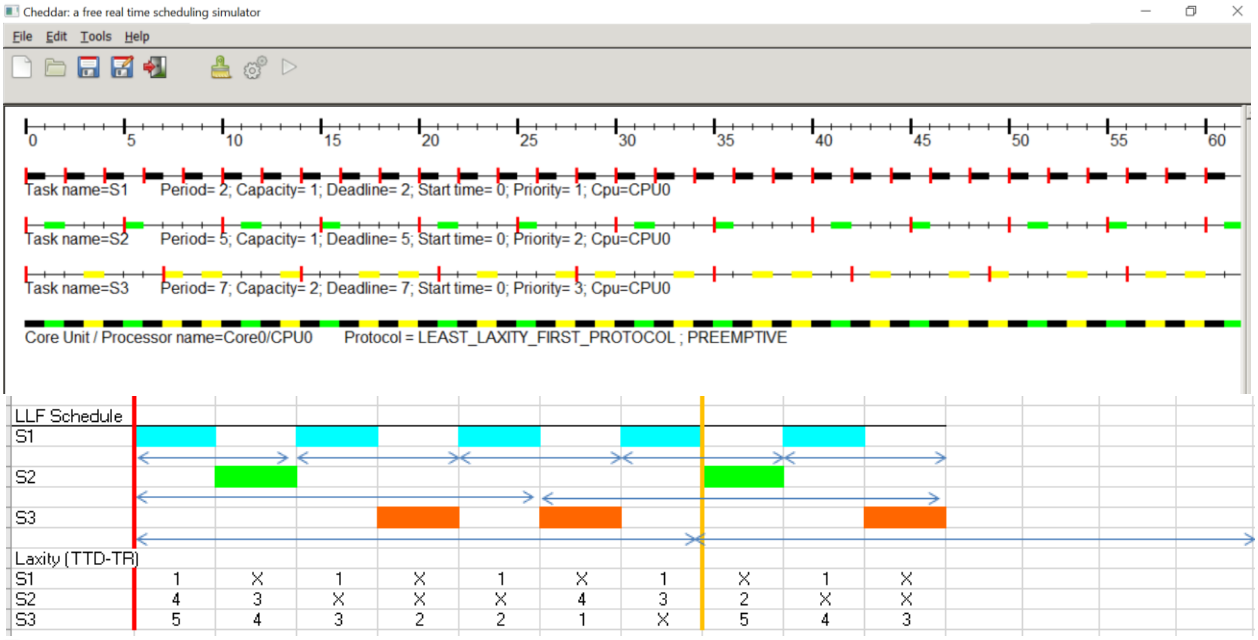


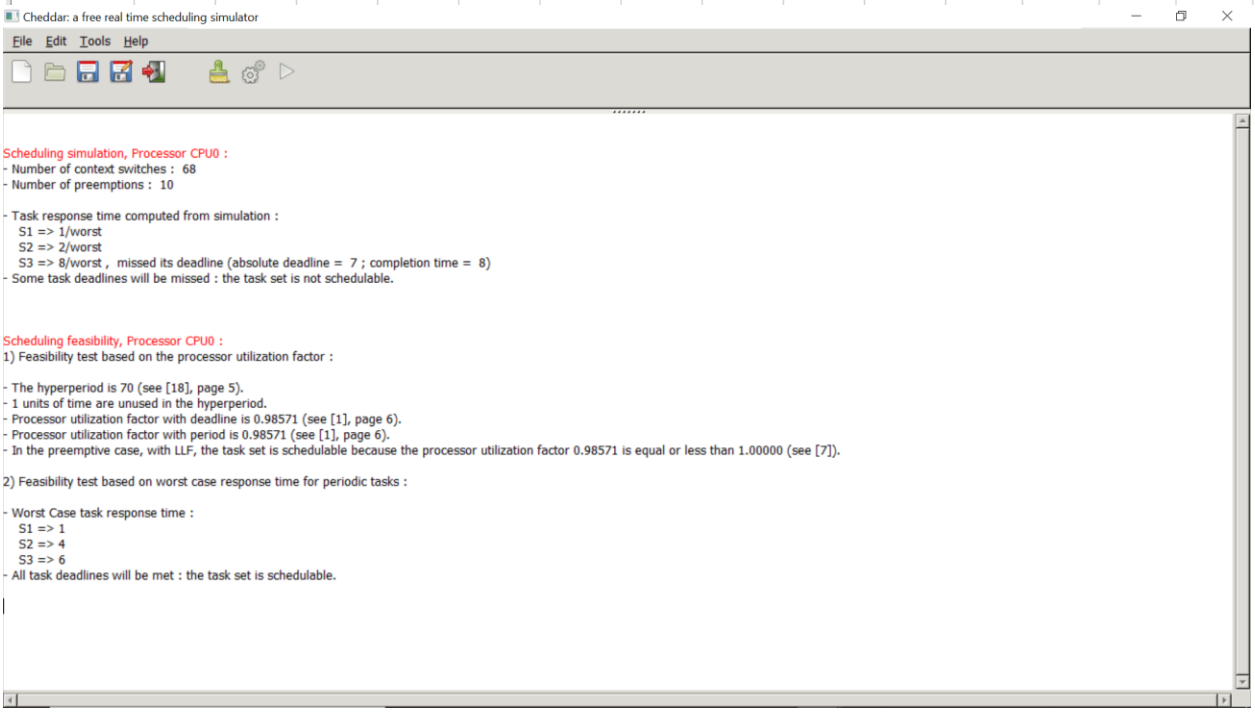
Example 1 LLF

LLF Scheduling: FEASIBLE (different results for Cheddar and timing diagram in excel)

-

- The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. Although there is difference in the timing diagrams. The simulation results in cheddar conclude that the task set is not schedulable and that some of the task deadlines are missed. Although, the scheduling feasibility tests in cheddar tool explains that the set of tasks is schedulable using EDF as the total utilization factor does not exceed 1.





SET 2:

S1	T1= 2	C1=1
S2	T2=5	C2=1
S3	T3=7	C3=1
S4	T4=13	C4=2

RM Scheduling: INFEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

The given set of tasks is not feasible to be scheduled using the RM policy as confirmed by all the three test methods, the code output, the timing diagram and the cheddar results. The total utilization factor is 99.67% while the RMLUB is 75.6828 %. Hence it fails the RMLUB test as well. The code outputs for the RM policy for completion tests and scheduling feasibility point test are given below, both of which provide output as INFEASIBLE. It is possible to schedule the given set of tasks using EDF anf LLF policies as provided in the timing diagrams and the cheddar results.

RASPI OUTPUT:

Completion Test

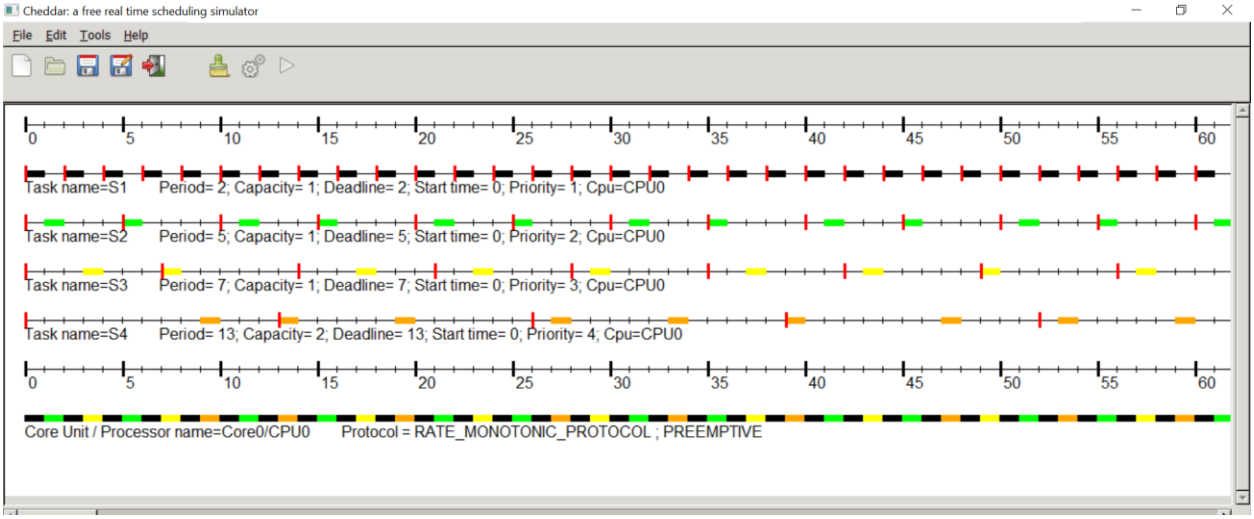
```
Ex-2 U=99.67% (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=1.000000, period=7.000000, utility_sum = 0.842857
for 3, wcet=2.000000, period=13.000000, utility_sum = 0.996703
utility_sum = 0.996703
LUB = 0.756828
RM LUB INFEASIBLE
```

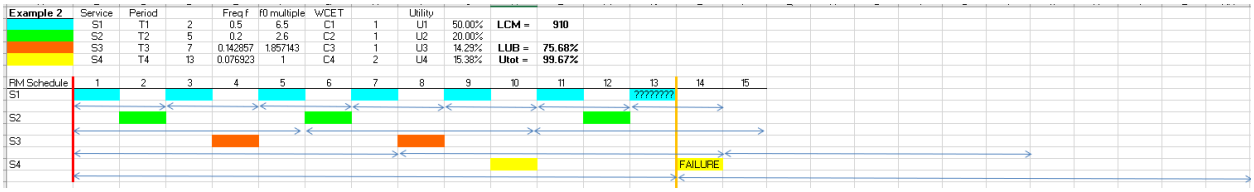
Scheduling Test

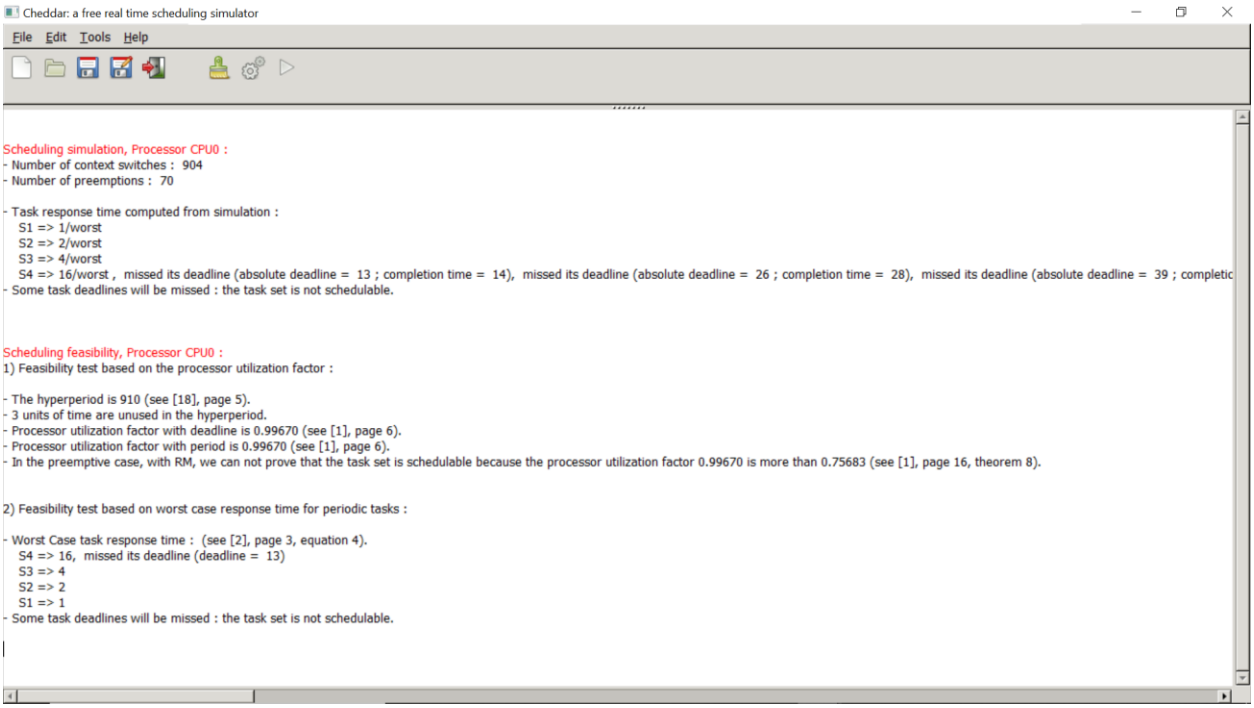
```
Ex-2 U=99.67% (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=1.000000, period=7.000000, utility_sum = 0.842857
for 3, wcet=2.000000, period=13.000000, utility_sum = 0.996703
utility_sum = 0.996703
LUB = 0.756828
RM LUB INFEASIBLE
```

Cheddar results

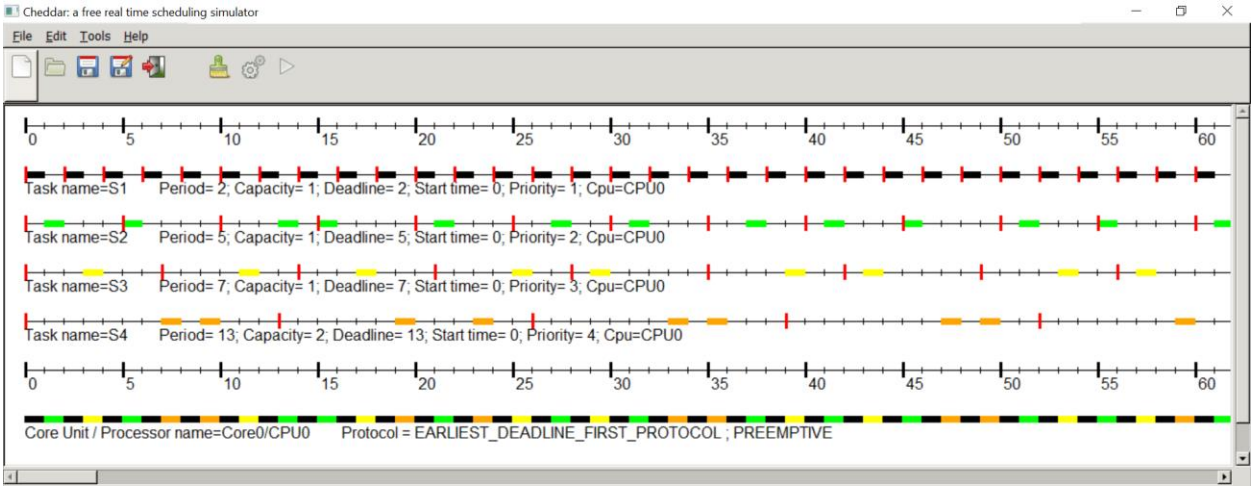
Example 2 RM : As depicted in the timing diagram, RM fails to schedule service S4 in the required deadline and hence fails to schedule the set of services. The cheddar results from scheduling simulation test provides the deadline and the completion time for the services that miss deadline.

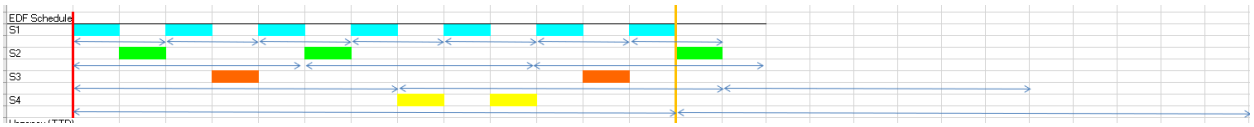


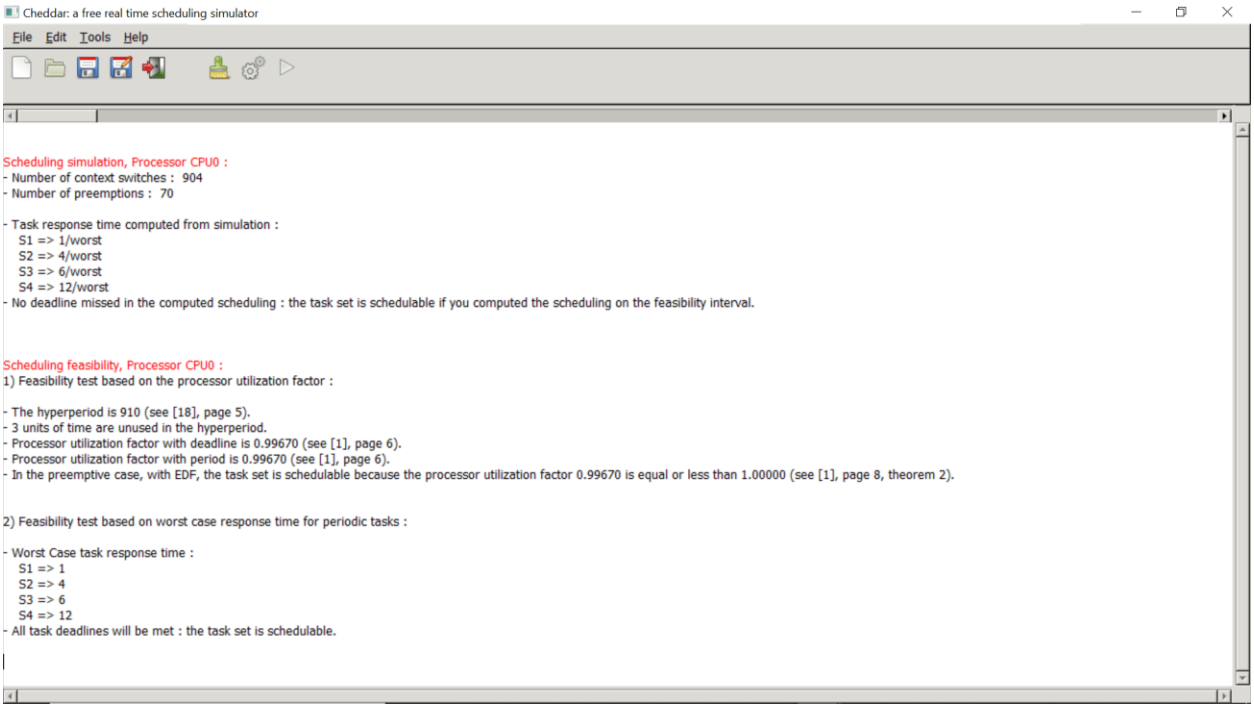




Example 2 EDF: The cheddar timing diagrams and the one given in the excel sheet match for the given policy. The set of tasks is schedulable using EDF proved by the cheddar scheduling simulation and scheduling feasibility results given below.

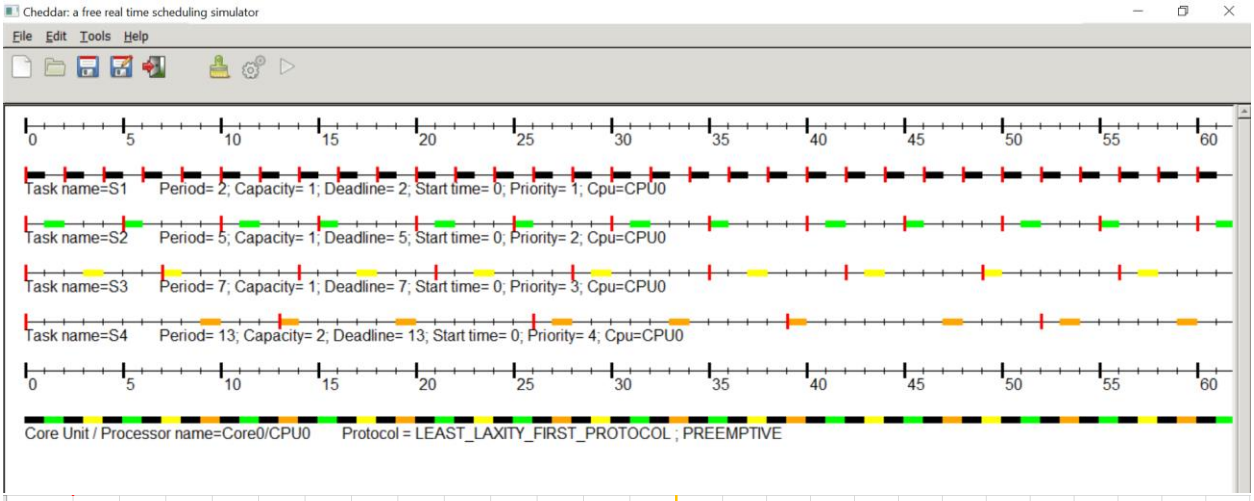




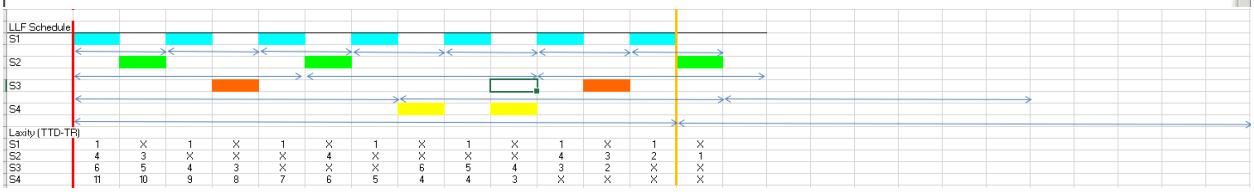


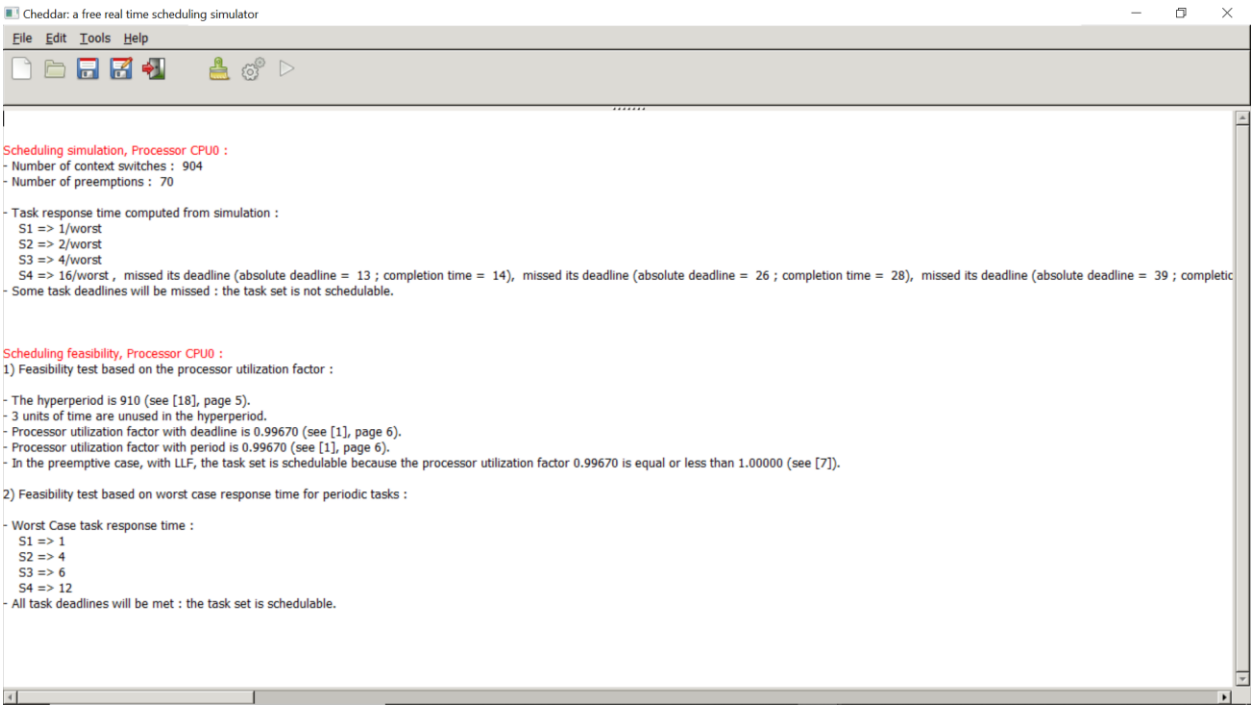
Example 2 LLF

- The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. The timing diagram in excel and the cheddar outputs match. The simulation results in cheddar conclude that the task set is not schedulable and that some of the task deadlines are missed. Although, the scheduling feasibility tests in cheddar tool explains that the set of tasks is schedulable using EDF as the total utilization factor does not exceed 1.



-





SET 3:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1= 3	C1=1
S2	T2=5	C2=2
S3	T3=15	C3=3

The given set of tasks is feasible using all the three methods- RM, EDF and LLF. Although, they do not pass the RMLUB test, the utilization factor is 93.33% while the RMLUB is 77.9763%. The completion test and the scheduling point feasibility test results confirm the set of tasks as feasible to be scheduled by RM. The timing diagram and the cheddar output confirm schedulability using EDF and LLF.

RASPI OUTPUT:

Completion Test

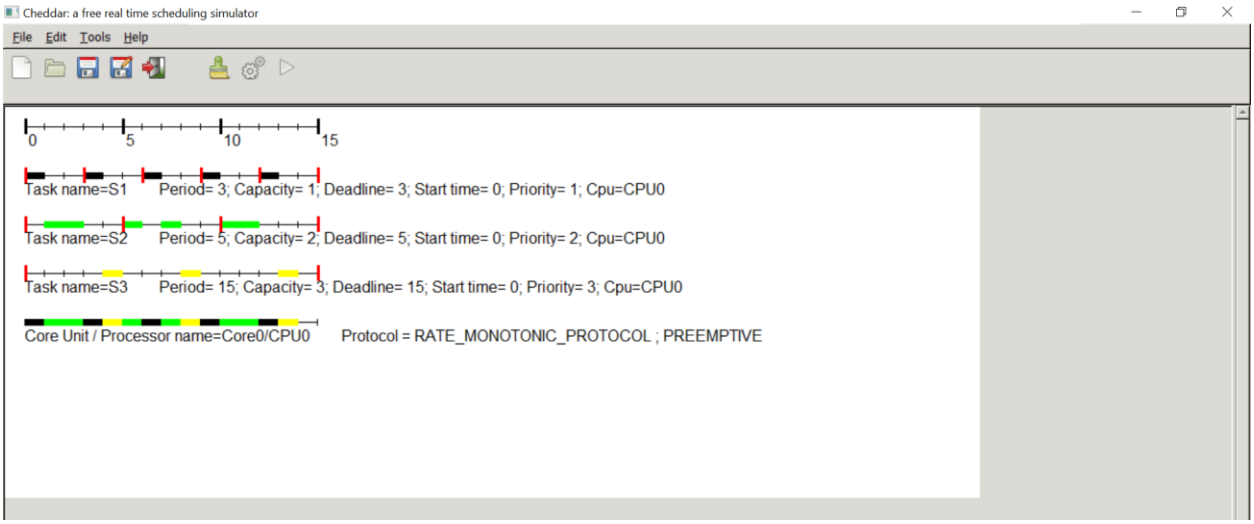
```
Ex-3 U=93.33% (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.733333
for 2, wcet=3.000000, period=15.000000, utility_sum = 0.933333
utility_sum = 0.933333
LUB = 0.779763
RM LUB INFEASIBLE
```

Scheduling Test

```
Ex-3 U=93.33% (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.733333
for 2, wcet=3.000000, period=15.000000, utility_sum = 0.933333
utility_sum = 0.933333
LUB = 0.779763
RM LUB INFEASIBLE
```

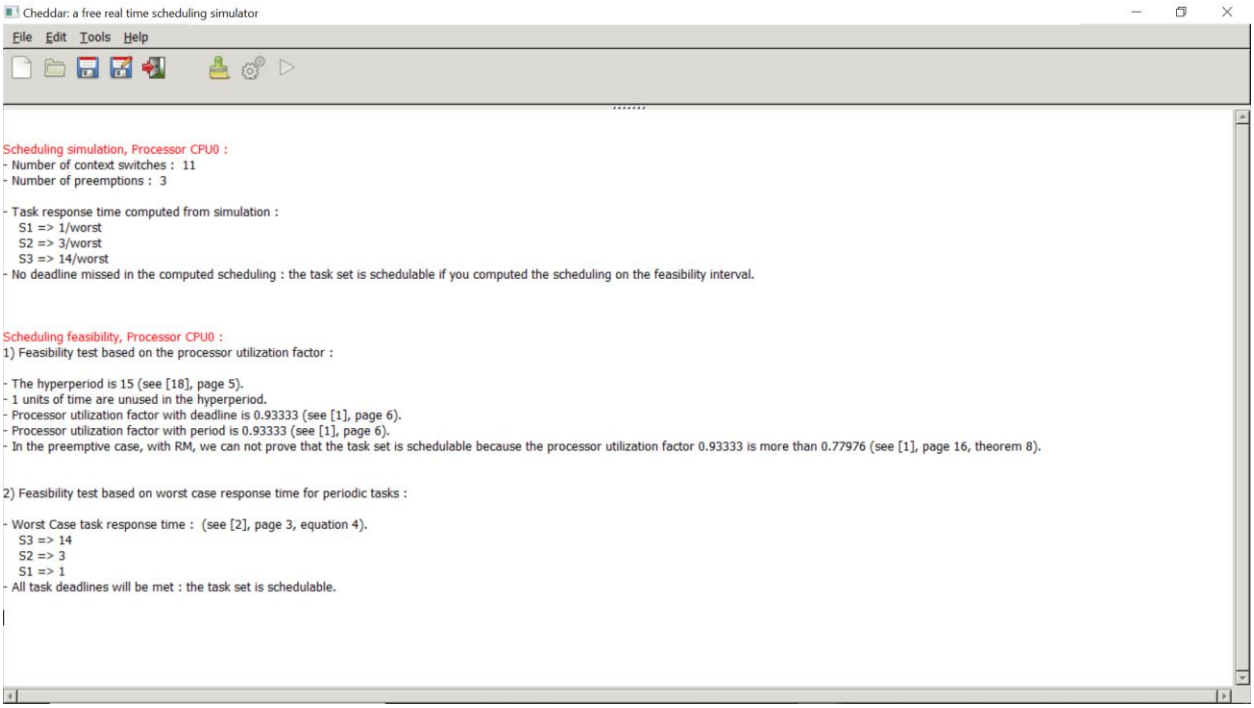
Cheddar results

Example 3 RM: The cheddar simulation results confirm that no deadline is missed and the services are satisfied appropriately. Th timing diagram also provides information regarding slack time. There is a margin of 6.67%, hence the results confirm that schedule is feasible but no comment can be made on the safety.

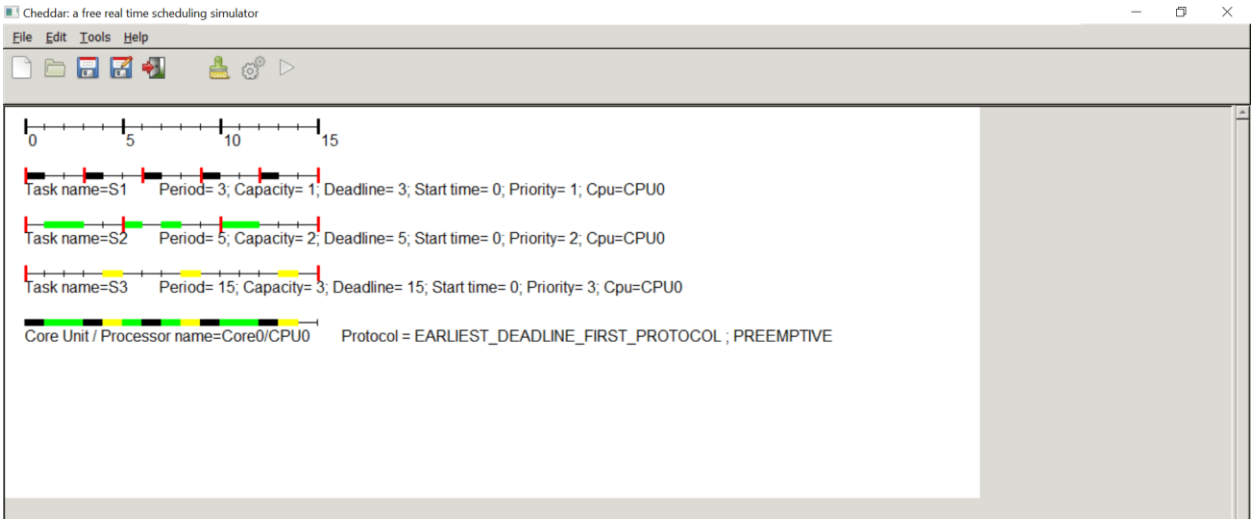


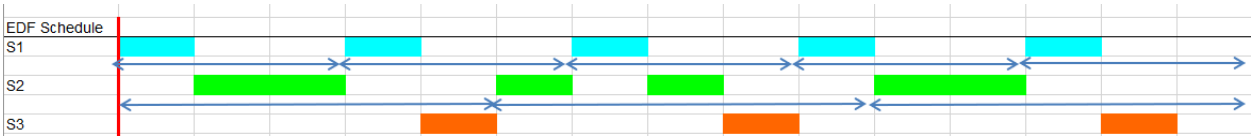
Example 3	Services	Freq f	f0 multiple	Period		WCET		Utility							
	S1	0.333333	5	T1	3	C1	1	U1	0.33	LCM =	15				
	S2	0.2	3	T2	5	C2	2	U2	0.4	LUB =	77.98%				
	S3	0.066667	1	T3	15	C3	3	U3	0.2	Utot =	93.33%	Slack	6.67%	Slack+U	100.00%

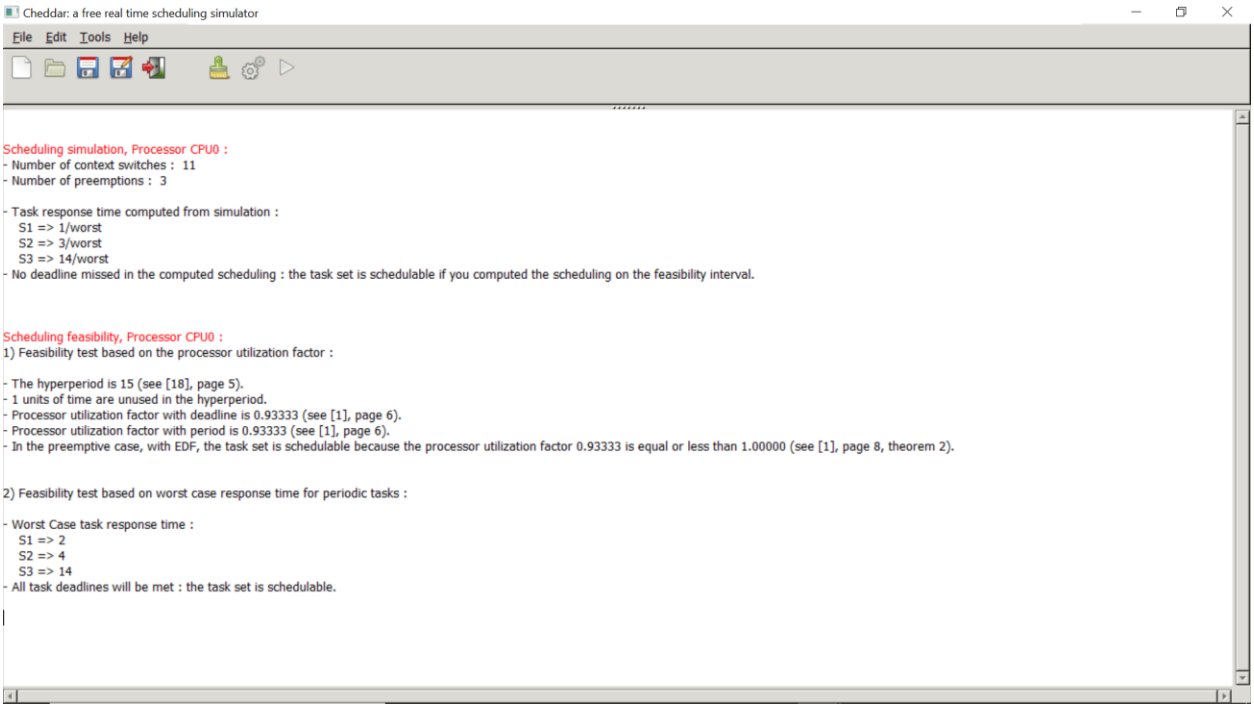
RM Schedule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1															Slack
S2															
S3															



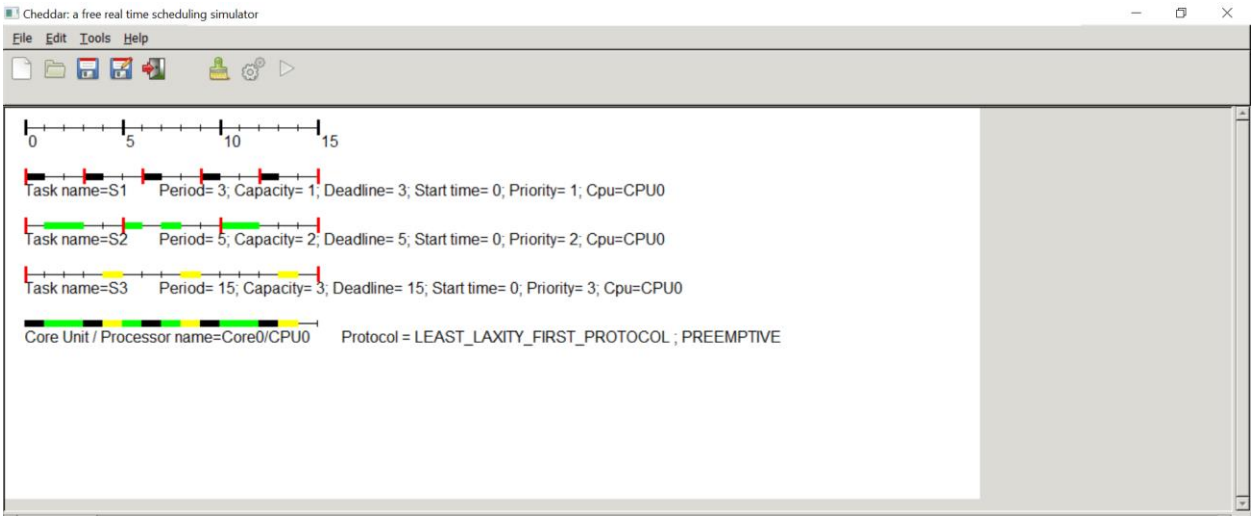
Example 3 EDF: The timing diagrams and the cheddar output matches and provide results that schedule is feasible for EDF policy.

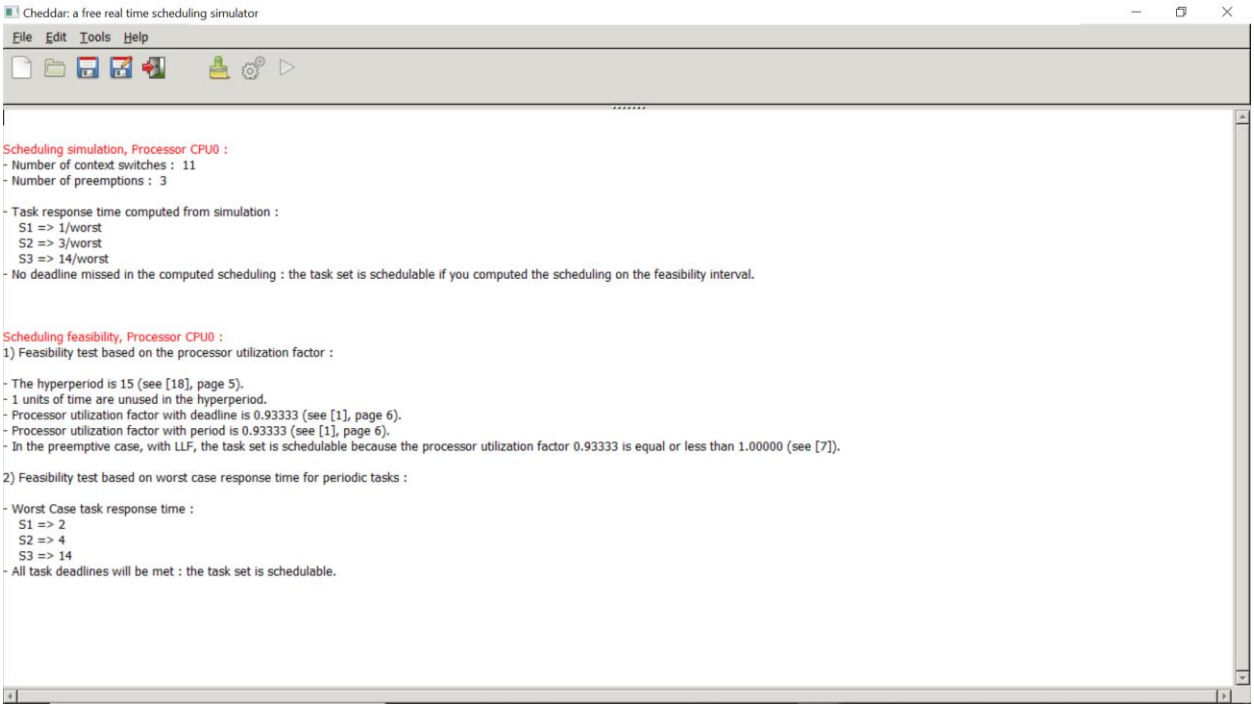






Example 3 LLF: The timing diagrams and the cheddar output matches and provide results that schedule is feasible for LLF policy.





SET 4:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1= 2	C1=1
S2	T2=4	C2=1
S3	T3=16	C3=4

The given set of tasks can be scheduled using all three policies. The utilization factor is 100% which is much higher than RMLUB- 77.9763% hence it fails the RMLUB test. Although, looking at the timing diagrams and the cheddar output, we can conclude that the schedule is feasible using RM policy. The completion test and scheduling point feasibility tests confirm that RM scheduling is feasible. The cheddar output for EDF and LLF tests suggest feasibility using both the policies.

RASPI OUTPUT:

Completion Test

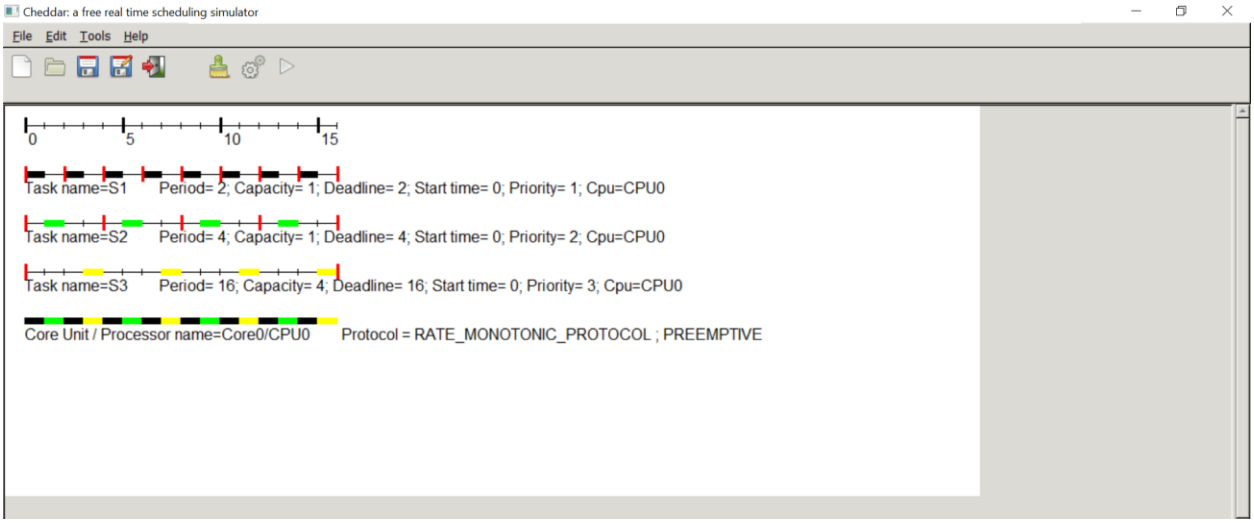
```
Ex-4 U=100.00% (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=4.000000, period=16.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

Scheduling Test

```
Ex-4 U=100.00% (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=4.000000, period=16.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

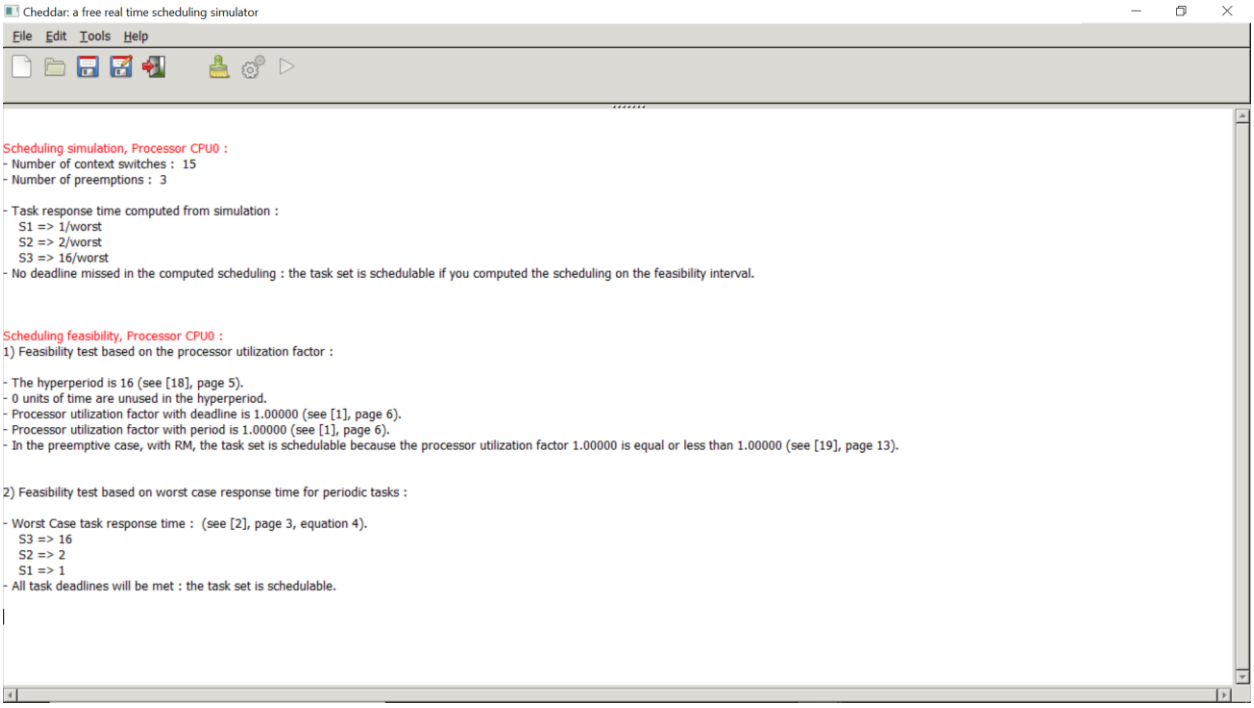
Cheddar results

Example 4 RM: The cheddar simulation results confirm that no deadline is missed, and the services are satisfied appropriately. There is 100% CPU utilization.

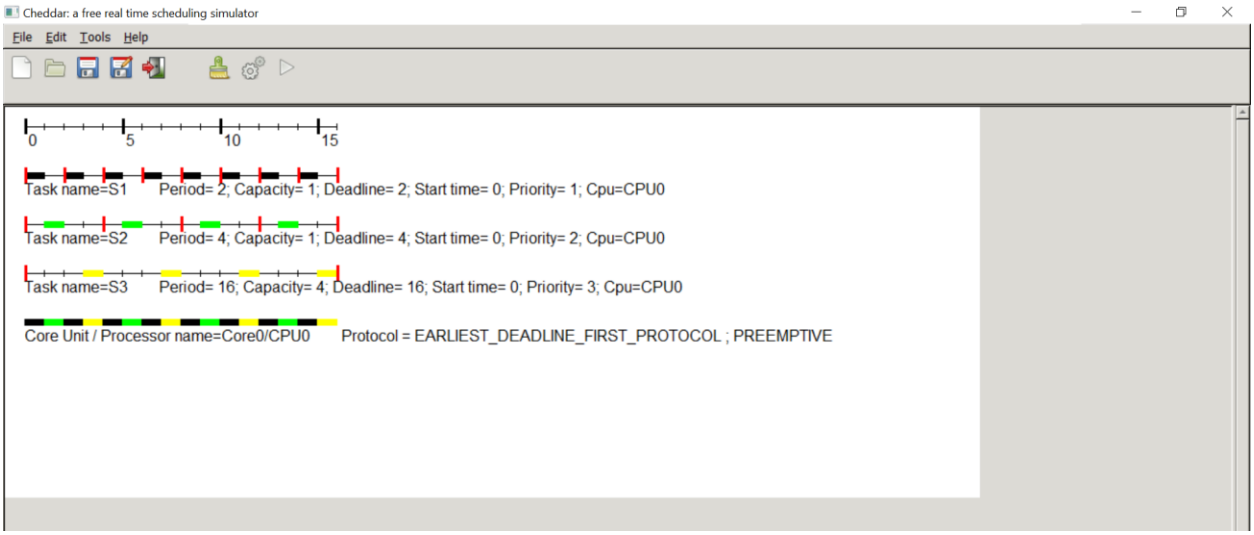


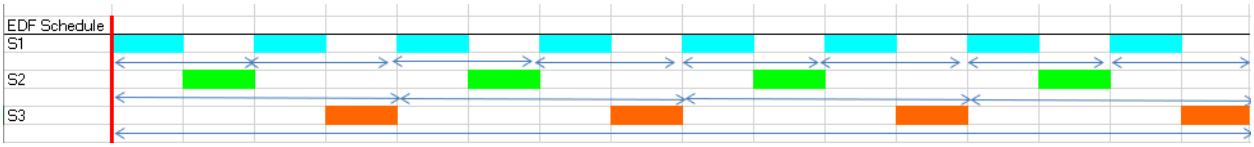
Example 4	Services	Freq f	f0 multiple	Period		WCET		Utility			
	S1	0.5	8	T1	2	C1	1	U1	0.5	LCM =	16
	S2	0.25	4	T2	4	C2	1	U2	0.25	LUB =	77.98%
	S3	0.0625	1	T3	16	C3	4	U3	0.25	Utot =	100.00%

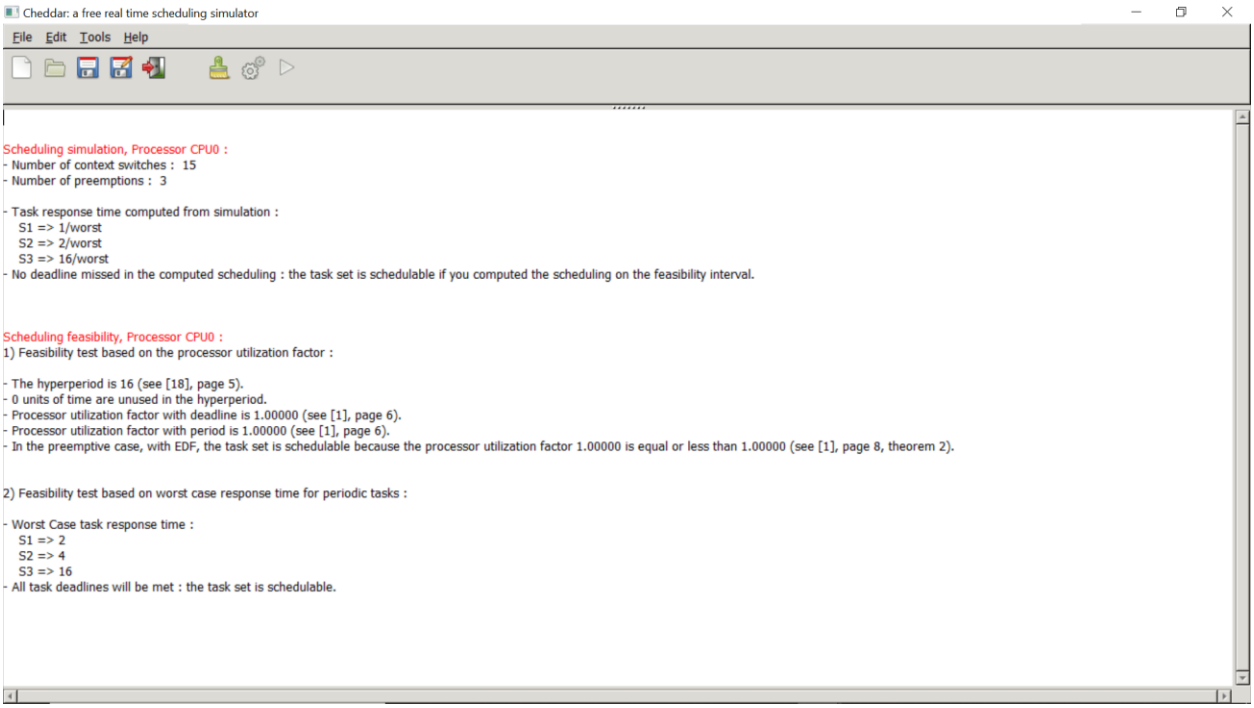
RM Schedule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S1																
S2																
S3																



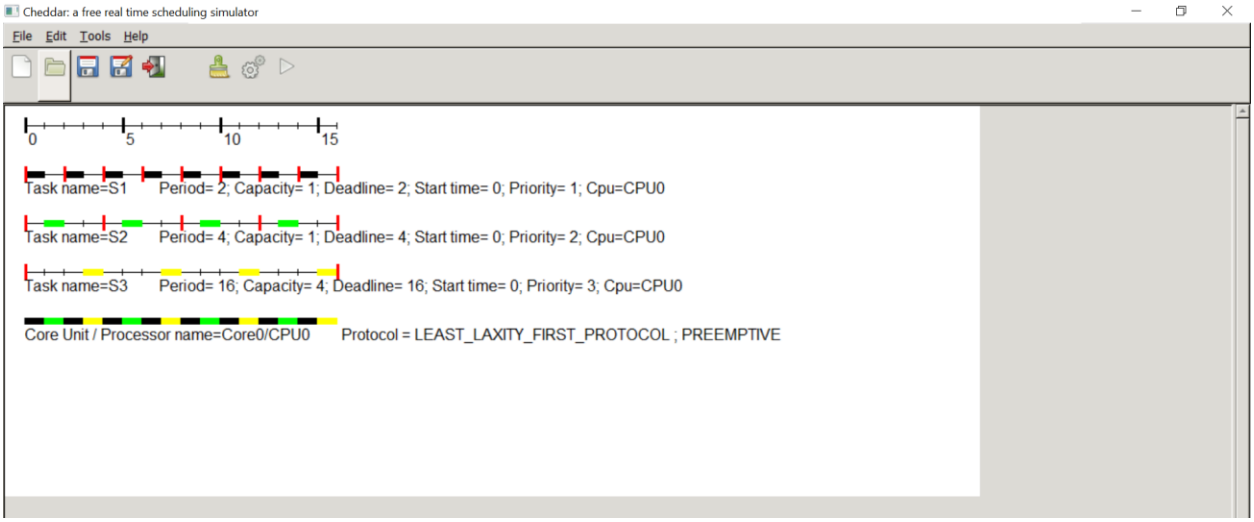
Example 4 EDF: The timing diagrams and the cheddar timing output match. The scheduling simulation and feasibility results confirm feasible output for EDF Scheduling for the given set of tasks.

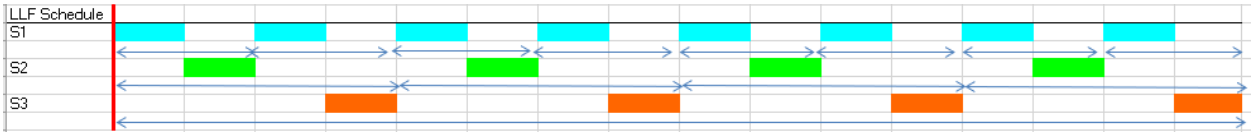


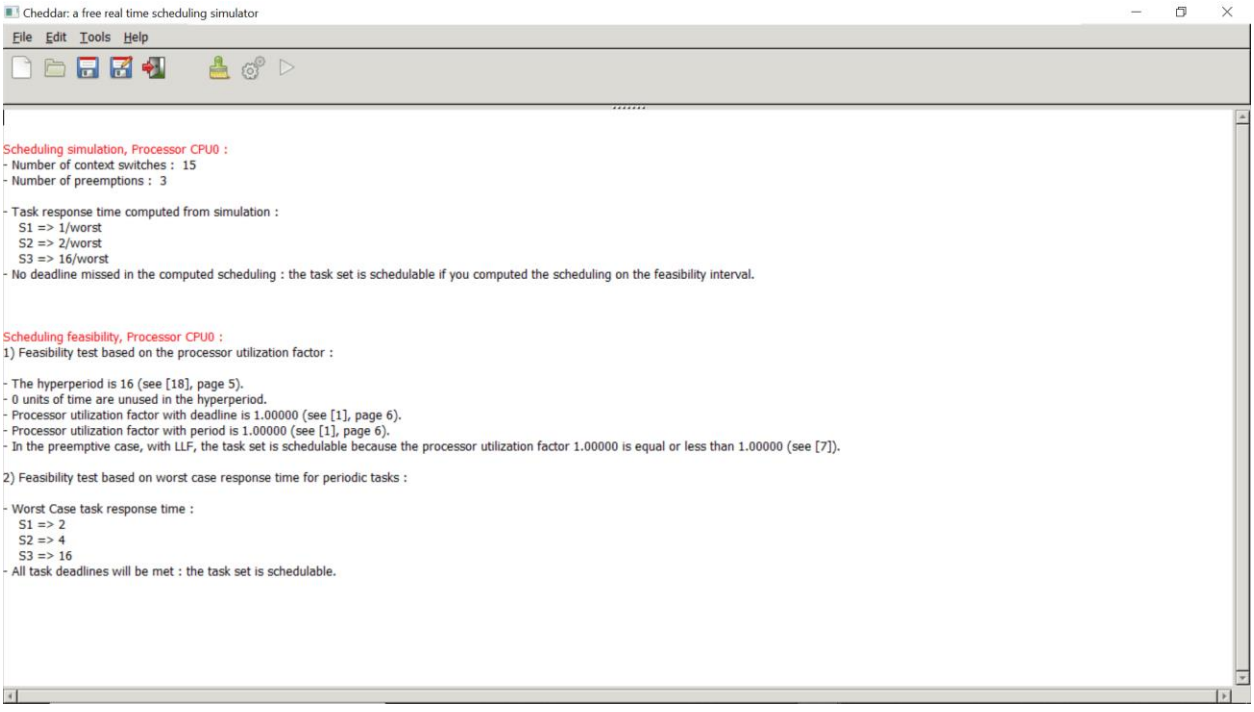




Example 4 LLF: The timing diagrams and the cheddar timing output match. The scheduling simulation and feasibility results confirm feasible output for LLF Scheduling for the given set of tasks (if computed scheduling on the feasibility interval).







b) Now, implement remaining examples [6 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".

The set of tasks taken into consideration
SET 5: (OVERLOAD Example)

S1	T1= 2	C1=1
S2	T2=4	C2=1
S3	T3=16	C3=5

RM Scheduling: INFEASIBLE
EDF Scheduling: INFEASIBLE
LLF Scheduling: INFEASIBLE

The given set of tasks is a critical example of OVERLOAD condition. The Utilizaition factor exceeds 100% and hence the set of task is not schedulable. None of the policies provide a feasible schedule to services all the task in the given deadline. The code outputs for completion and scheduling point feasibility tests confirm the same. Cheddar timing diagram and results match these results.

RASPI OUTPUT:

Completion Test

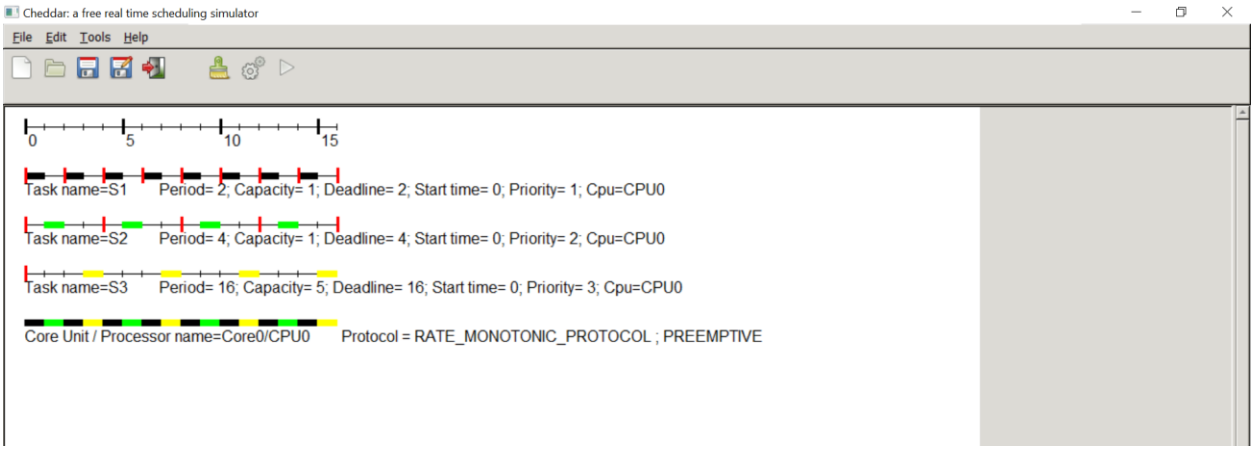
```
Ex-5 U=106.25% (C1=1, C2=1, C3=5; T1=2, T2=4, T3=16; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=5.000000, period=16.000000, utility_sum = 1.062500
utility_sum = 1.062500
LUB = 0.779763
RM LUB INFEASIBLE
```

Scheduling Test

```
Ex-5 U=106.25% (C1=1, C2=1, C3=5; T1=2, T2=4, T3=16; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=5.000000, period=16.000000, utility_sum = 1.062500
utility_sum = 1.062500
LUB = 0.779763
RM LUB INFEASIBLE
```

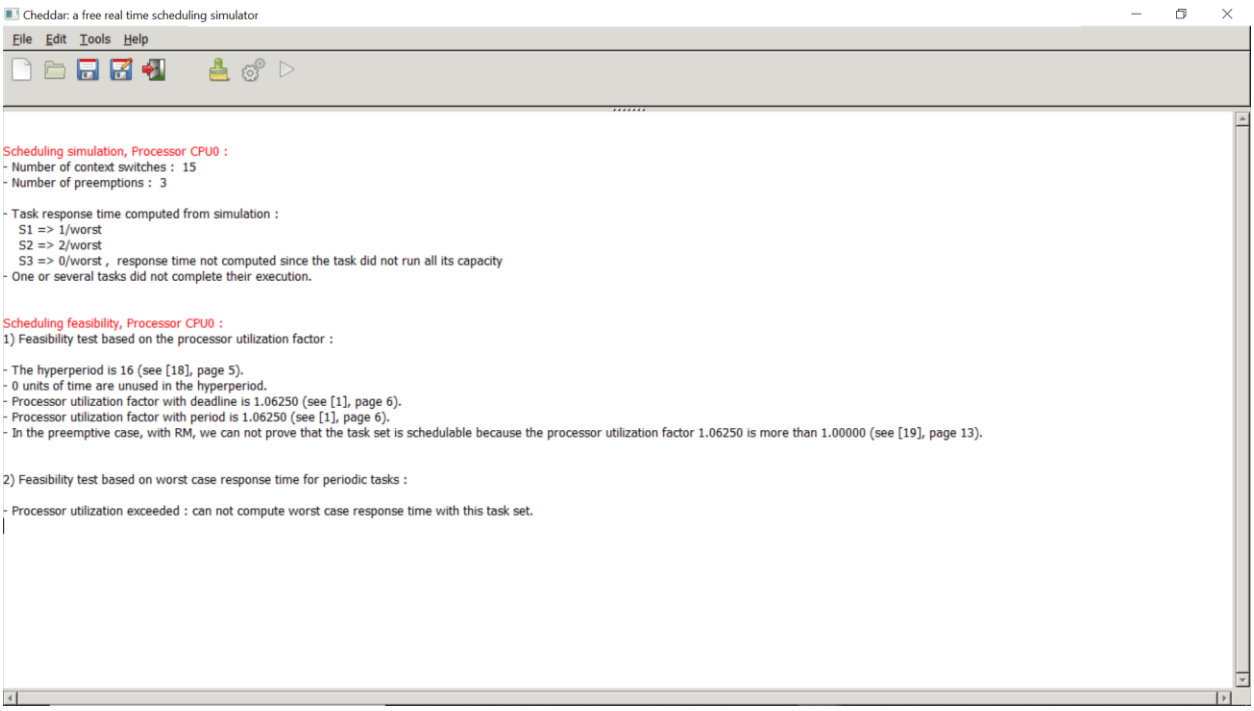
Cheddar results:

Example 5 RM: The deadline for service S3 is missed in the given time interval and hence RM fails to schedule the given set of tasks. The simulation results clearly mention, one or several tasks did not complete their execution. The scheduling feasibility results mention “Processor utilization exceeded” and hence the set of tasks is not feasible.

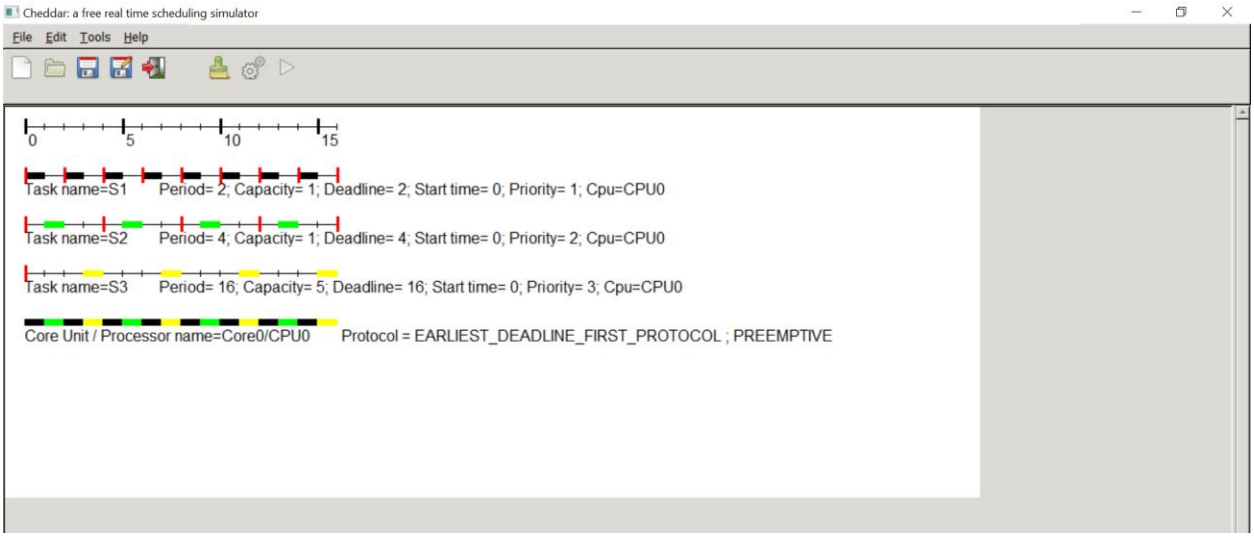


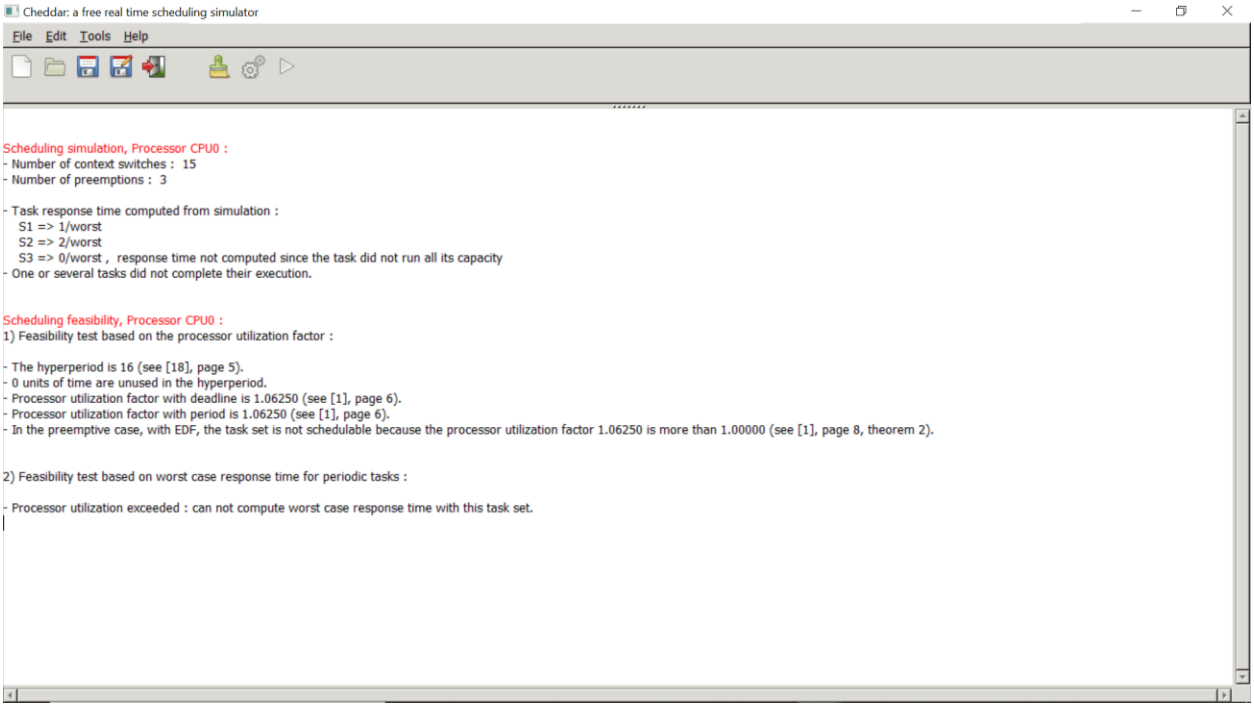
Example 4	Services	Freq f	f0 multiple	Period		WCET		Utility			
	S1	0.5	8	T1	2	C1	1	U1	0.5	**LCM =**	**16**
	S2	0.25	4	T2	4	C2	1	U2	0.25	**LUB =**	**77.98%**
	S3	0.0625	1	T3	16	C3	5	U3	0.3125	**Utot =**	**106.25%**

RM Schedule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S1																
S2																
S3																MISS

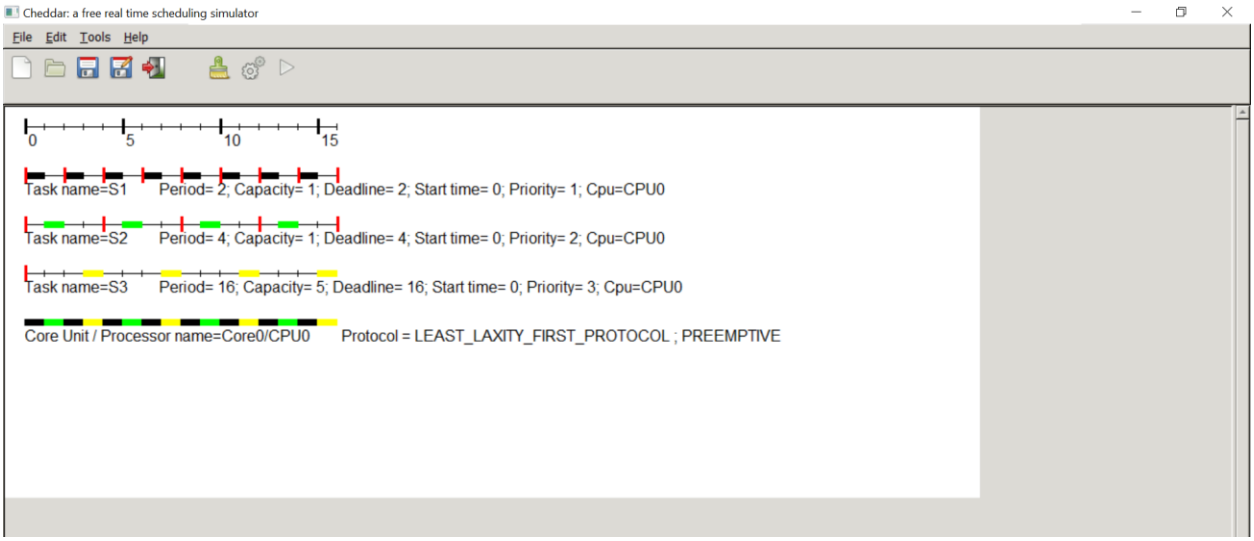


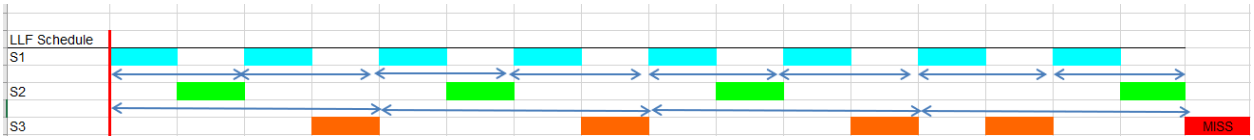
Example 5 EDF: The results are similar to RM policy (not in terms of timing diagram) as the processor utilization exceeds the limit hence EDF fails to schedule the set of tasks.

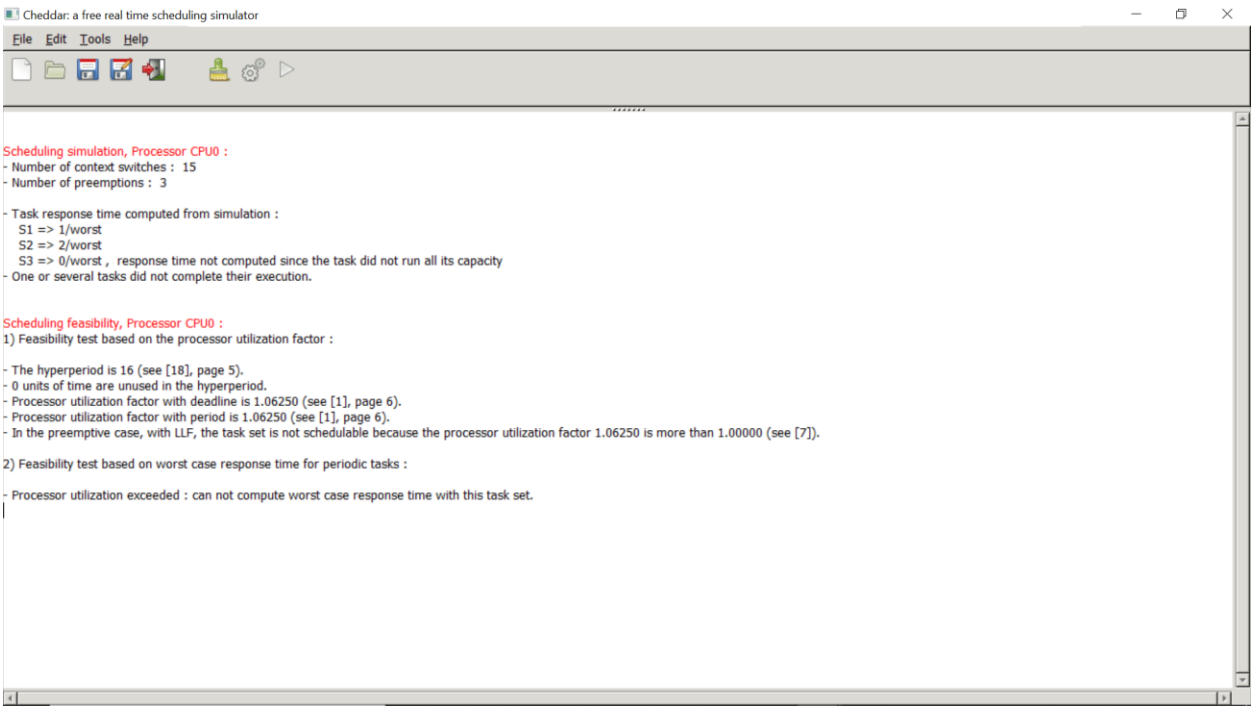




Example 5 LLF: The timing diagram in excel meets the cheddar timing output. The simulation and feasibility tests result clearly suggest the schedule is INFEASIBLE.







SET 6:

S1	T1=2	C1=1
S2	T2=5	C2=2
S3	T3=10	C3=1

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

The given set of task is feasible using all the three policies. The total utilization factor exceeds the RMLUB (U = 100% while RMLUB = 77.9763%). The code results for completion tests and scheduling feasibility point tests suggest that RM schedule is feasible. The cheddar and timing diagram results are provided below.

RASPI OUTPUT:

Completion Test

```
Ex-6 U=100.00% (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.900000
for 2, wcet=1.000000, period=10.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

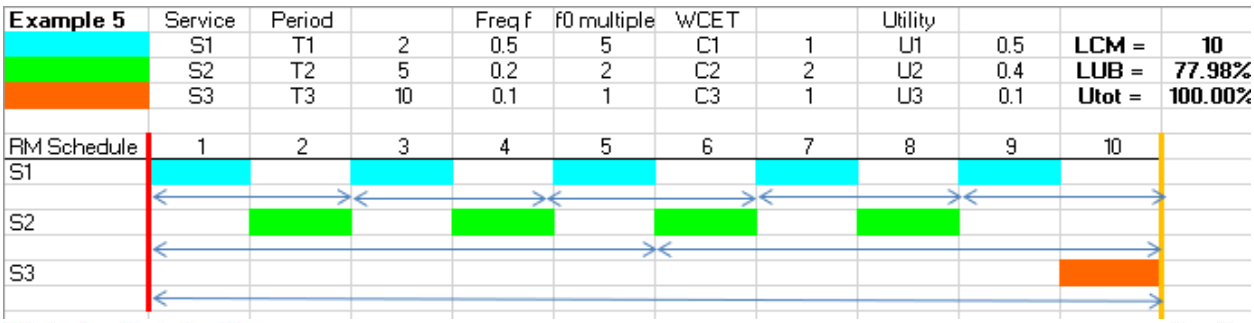
Scheduling Test

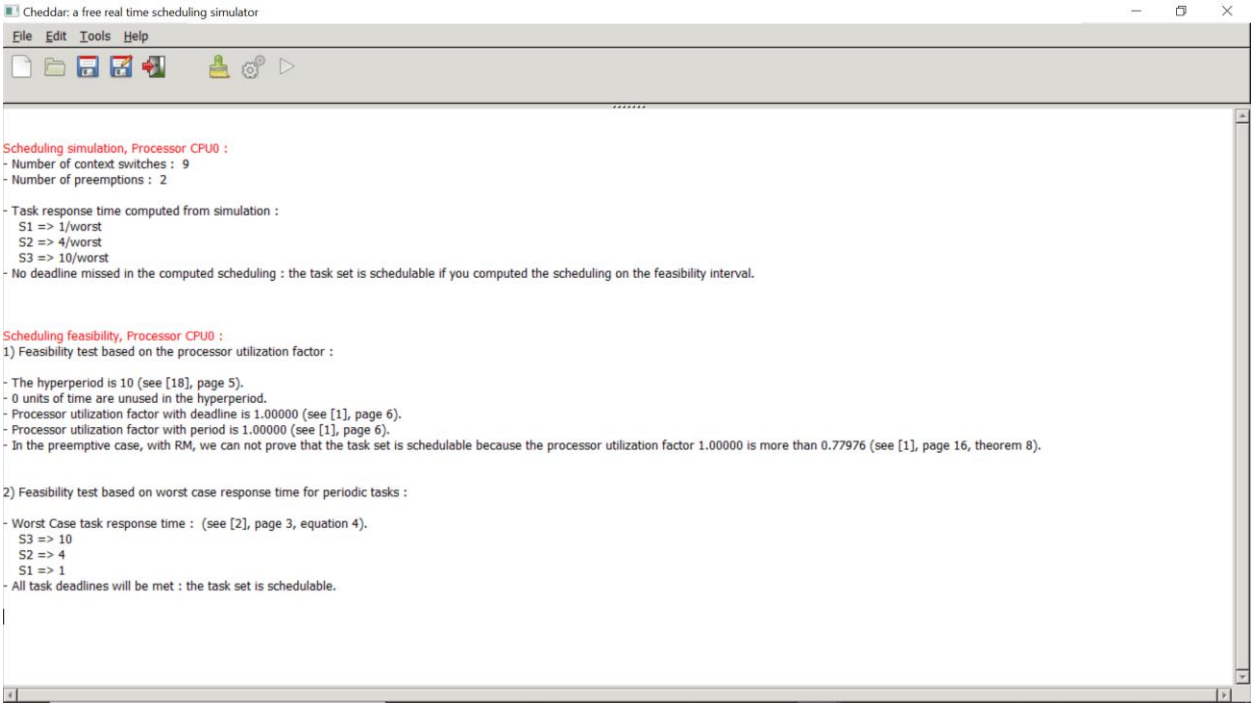
```
Ex-6 U=100.00% (C1=1, C2=2, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.900000
for 2, wcet=1.000000, period=10.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

Cheddar results:

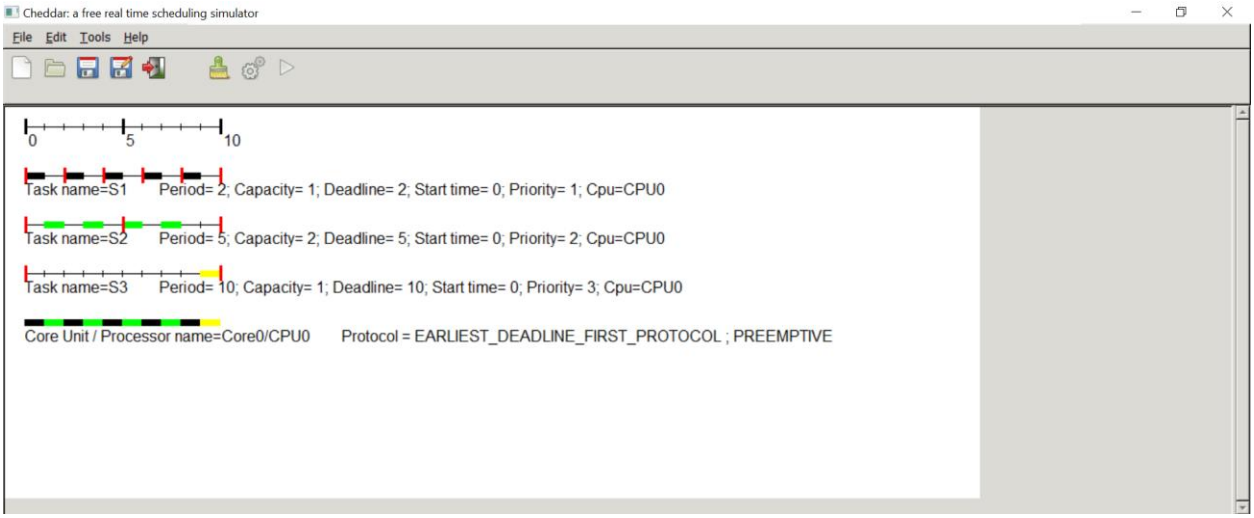
Example 6 RM: Utot exceeds RMLUB and hence there is no guarantee for the safety of current schedule. But, the timing diagrams from the excel and Cheddar results suggest that it is feasible to schedule given set of services using RM policy.

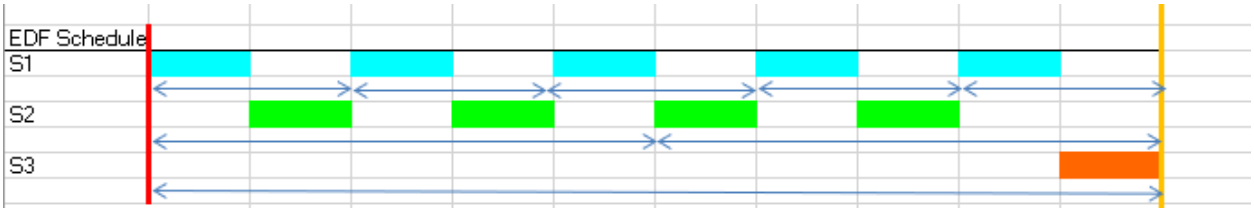
Example 5	Service	Period		Freq f	f0 multiple	WCET		Utility			
	S1	T1	2	0.5	5	C1	1	U1	0.5	LCM =	10
	S2	T2	5	0.2	2	C2	2	U2	0.4	LUB =	77.98%
	S3	T3	10	0.1	1	C3	1	U3	0.1	Utot =	100.00%

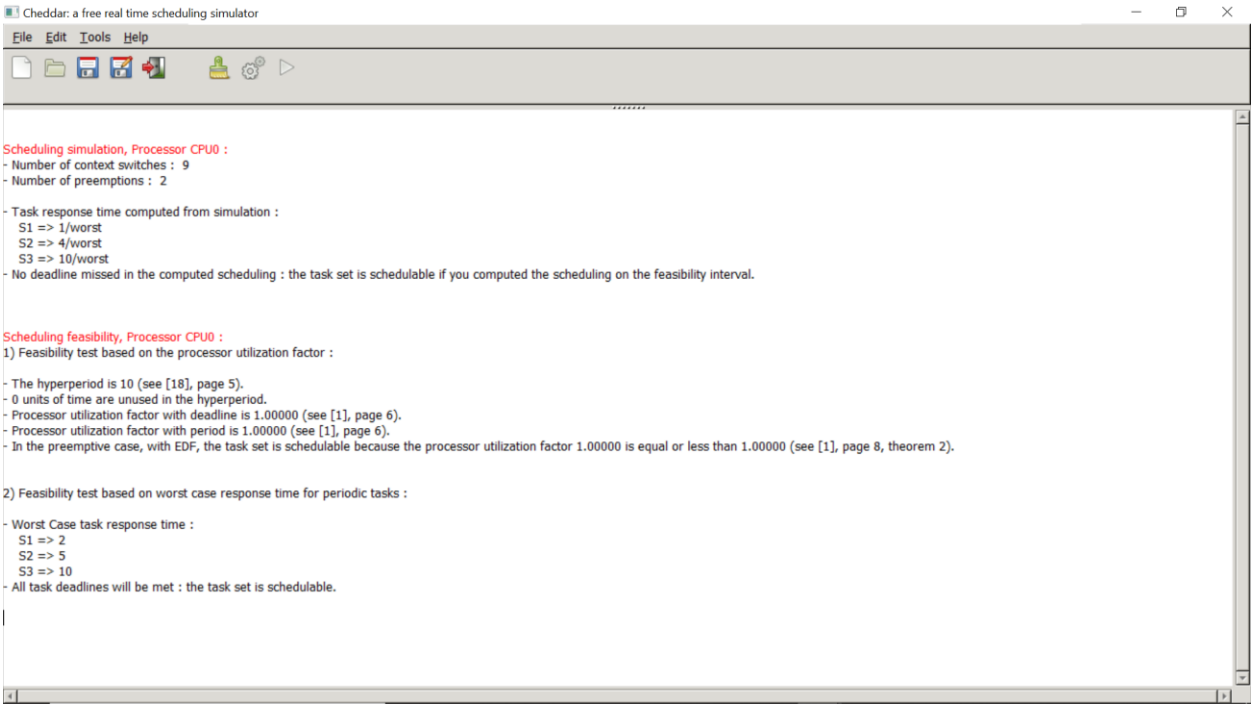




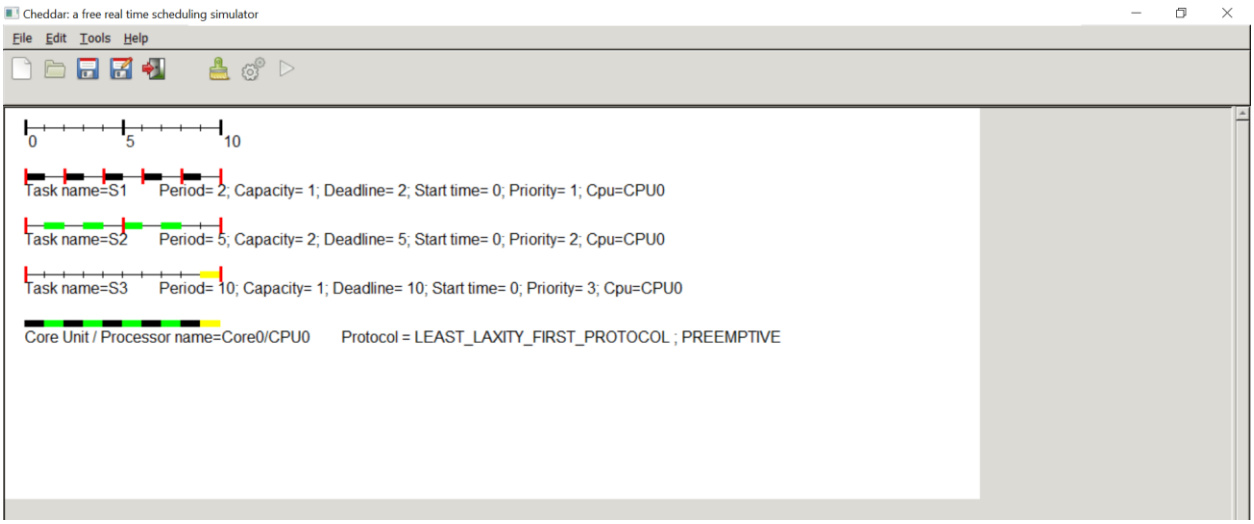
Example 6 EDF: Timing diagrams match for excel and cheddar tool. The simulation and feasibility point results provide task response time and suggest that the schedule is feasible.

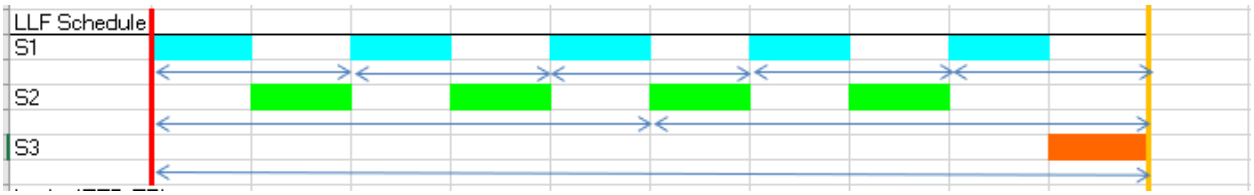


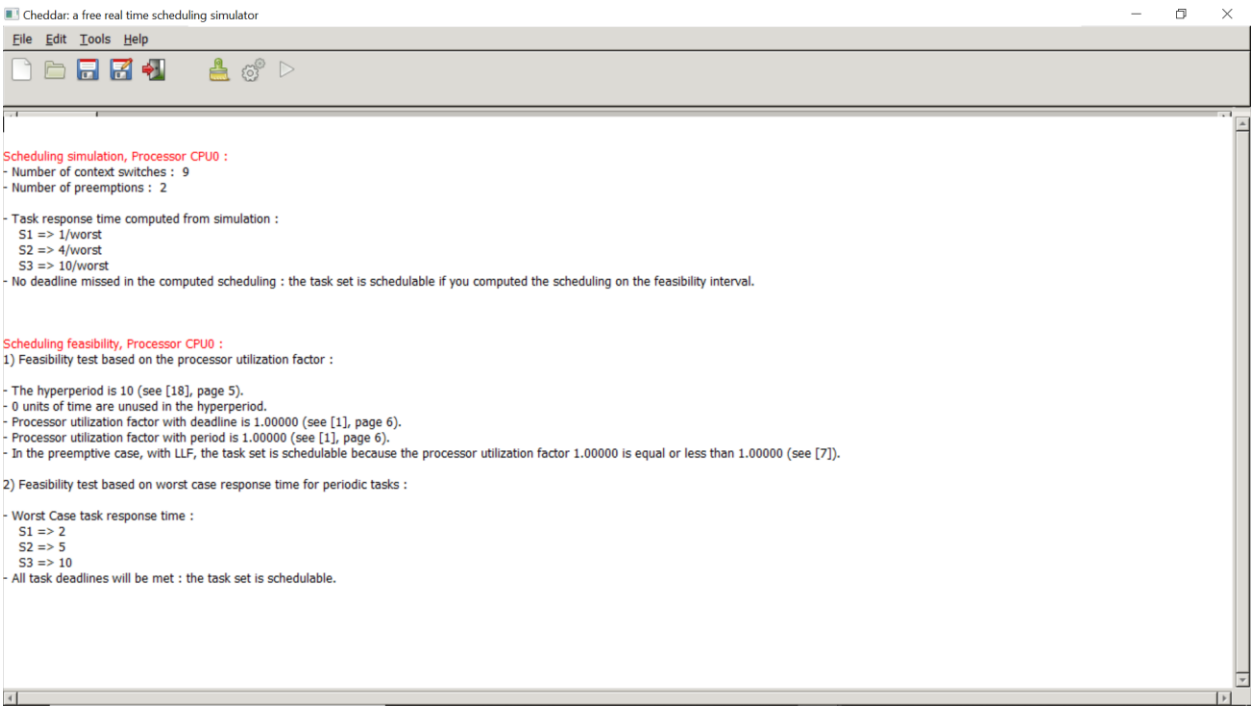




Example 6 LLF: Timing diagrams and cheddar simulation+ feasibility results suggest schedule is feasible using LLF.







SET 7:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1=3	C1=1
S2	T2=5	C2=2
S3	T3=15	C3=4

RMLUB = 77.9763% while U= 100% hence the RMLUB test is not satisfied. Although timing diagrams and cheddar results suggest schedule feasibility using RM policy. The completion test and the scheduling feasibility point test also provide “FEASIBLE” output. EDF and LLF also provide feasible scheduling for given set of tasks.

RASPI OUTPUT:

Completion Test

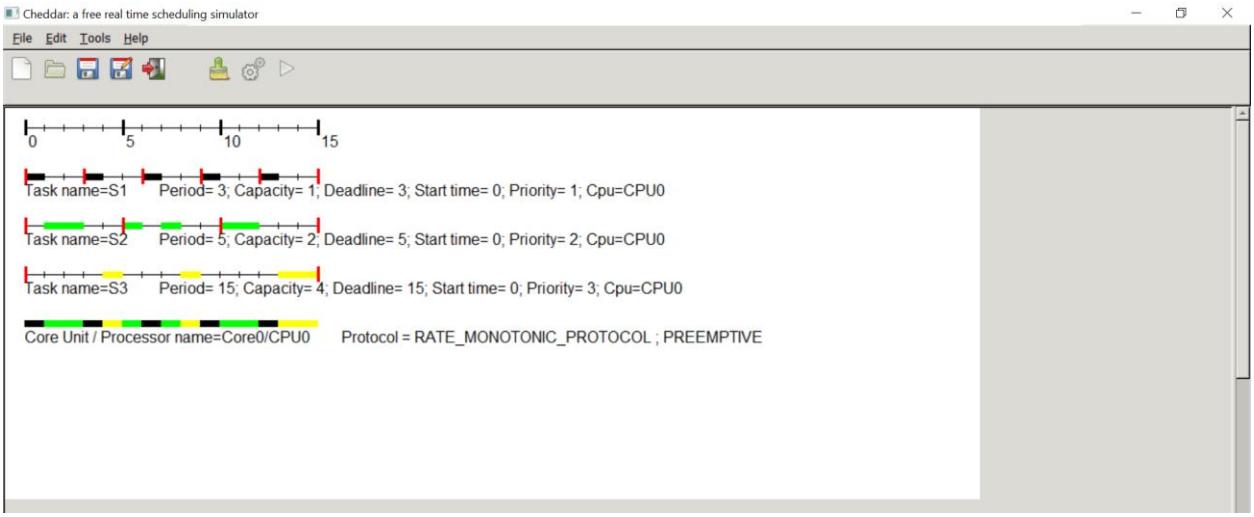
```
Ex-7 U=100.00% (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.733333
for 2, wcet=4.000000, period=15.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

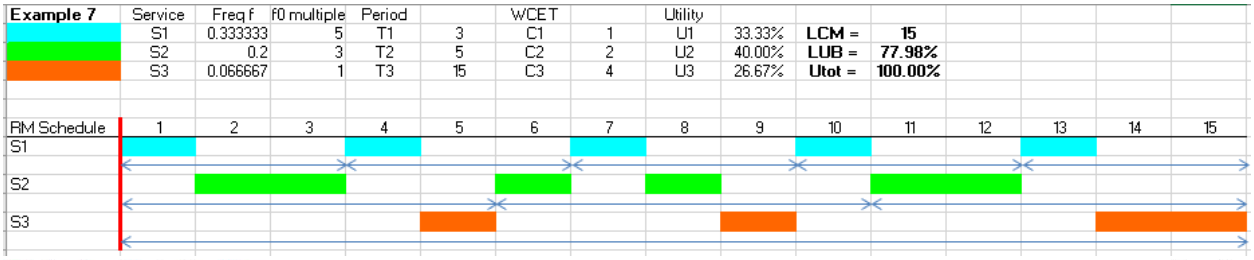
Scheduling Test

```
Ex-7 U=100.00% (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=5.000000, utility_sum = 0.733333
for 2, wcet=4.000000, period=15.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

Cheddar Results:

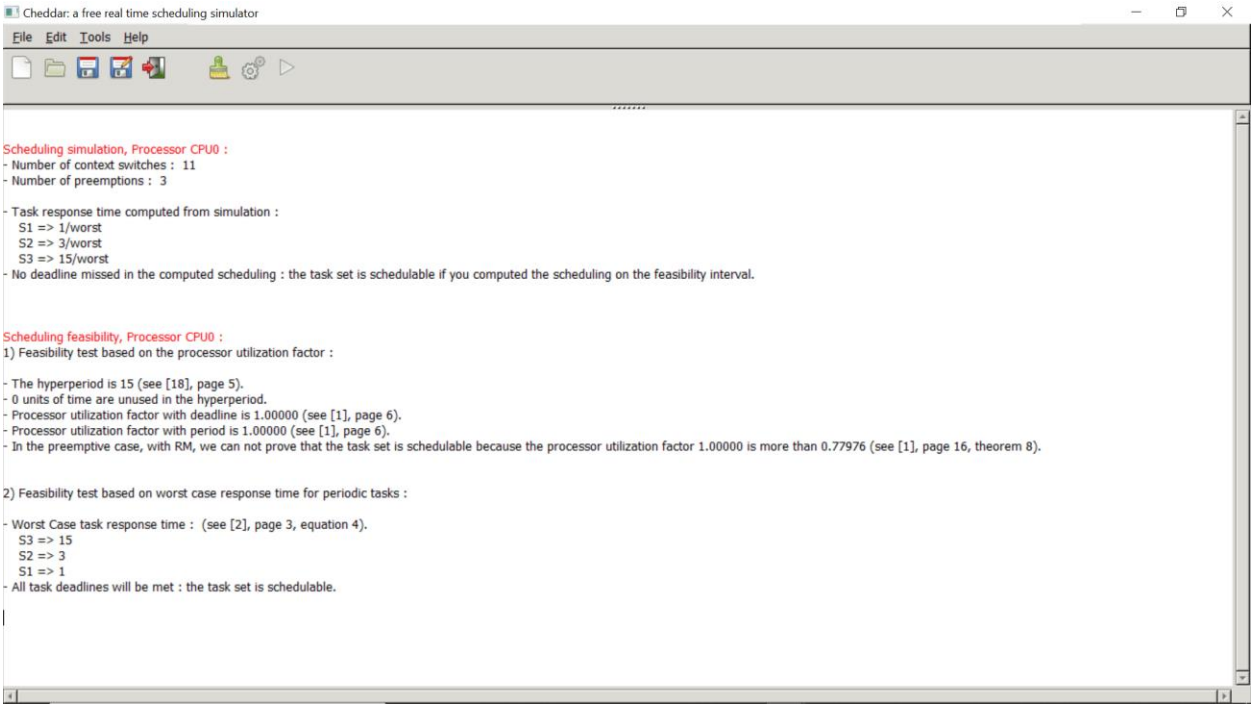
Example 7 RM: The schedule does not pass RMLUB Test but the timing diagrams in excel and cheddar feasibility results suggest that we can schedule the given set of tasks using RM policy.



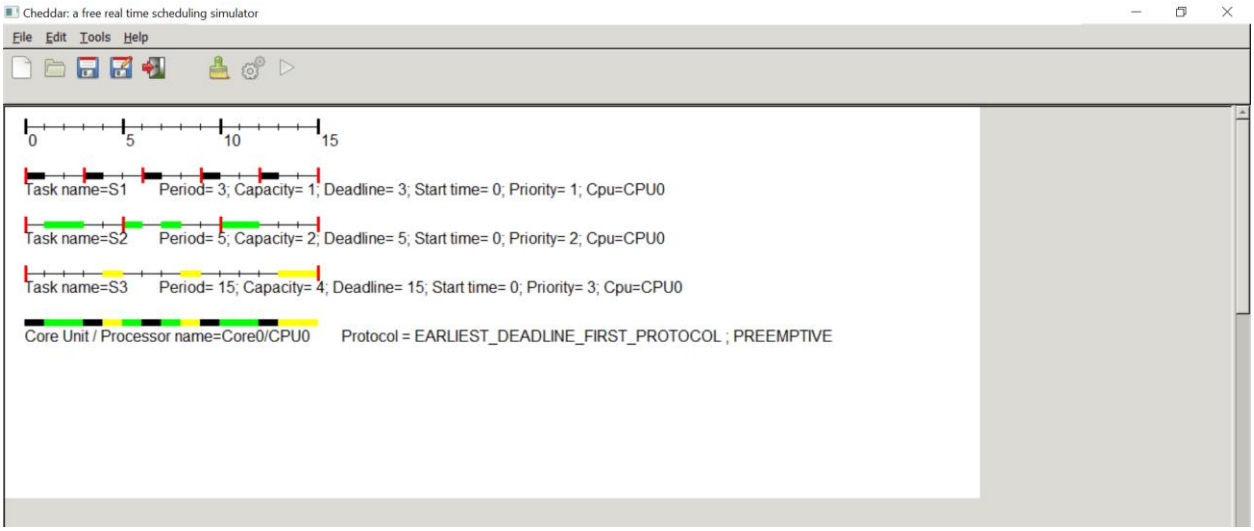


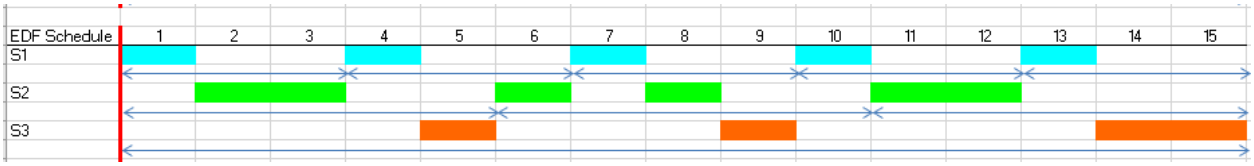
Example 7	Service	Freq f	f0 multiple	Period		WCET		Utility			
	S1	0.333333	5	T1	3	C1	1	U1	33.33%	LCM =	15
	S2	0.2	3	T2	5	C2	2	U2	40.00%	LUB =	77.98%
	S3	0.066667	1	T3	15	C3	4	U3	26.67%	Utot =	100.00%

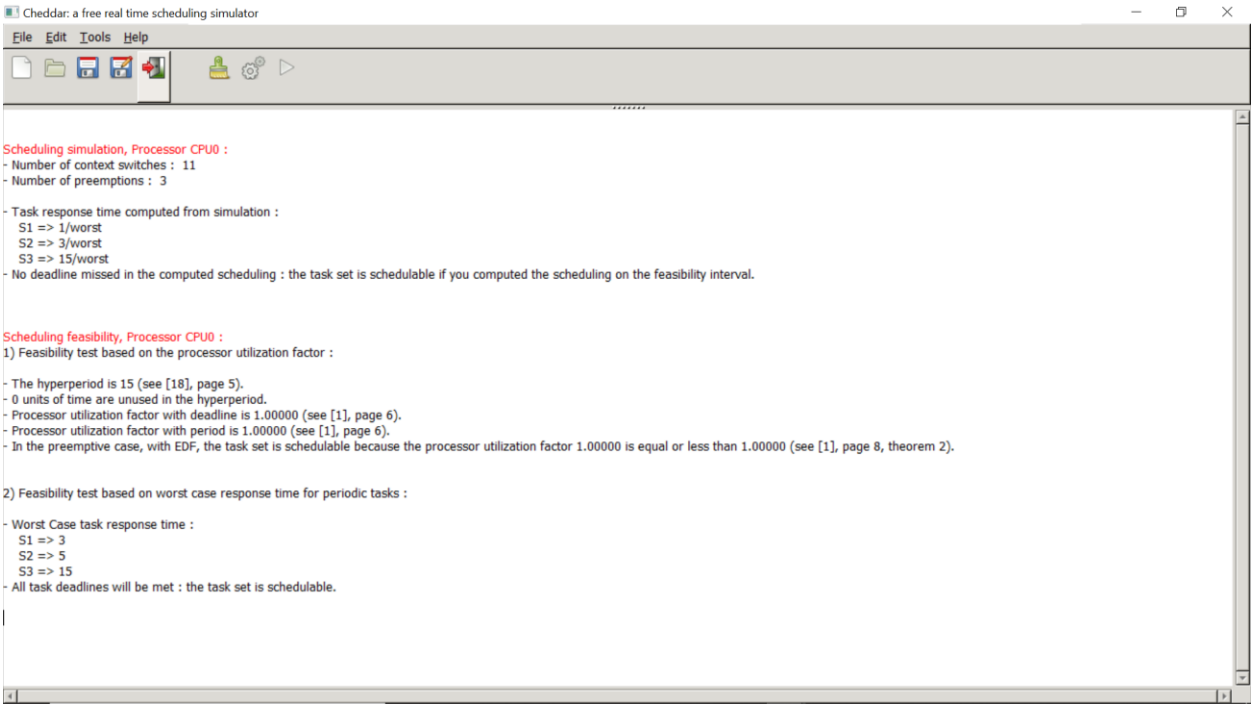
RM Schedule	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1															
S2															
S3															



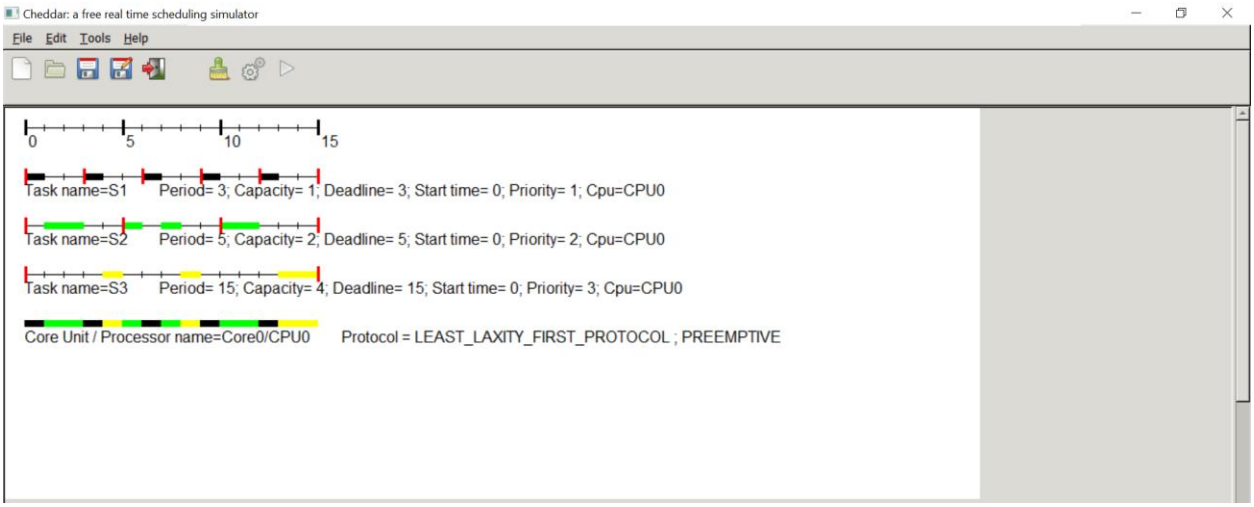
Example 7 EDF: Matching timing diagrams for excel and Cheddar, scheduling simulation and scheduling feasibility test suggest feasible schedule using EDF.

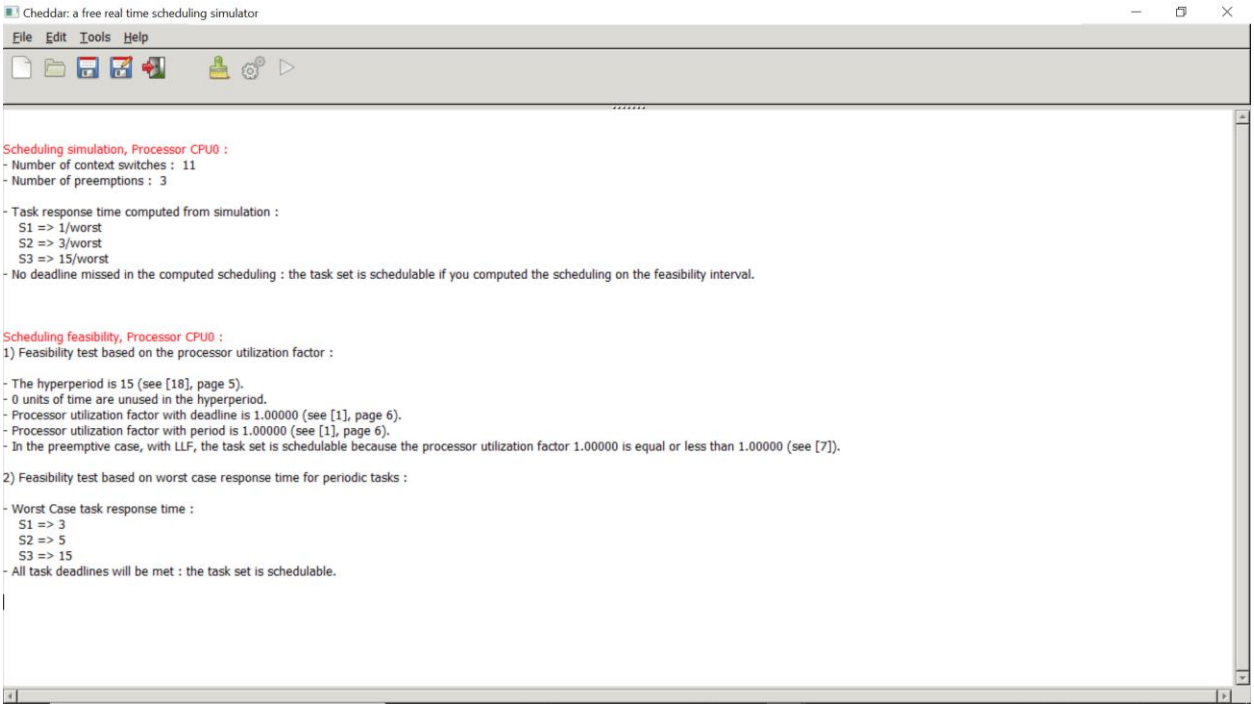






Example 7 LLF: -The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. Although there is difference in the timing diagrams.





SET 8:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1=6	C1=1
S2	T2=8	C2=2
S3	T3=12	C3=4
S4	T4=24	C4=6

The given set of task is feasible using all the three policies. The total utilization factor exceeds the RMLUB (U = 100% while RMLUB = 75.6828%). The code results for completion tests and scheduling feasibility point tests suggest that RM schedule is feasible. The cheddar and timing diagram results are provided below.

Completion Test

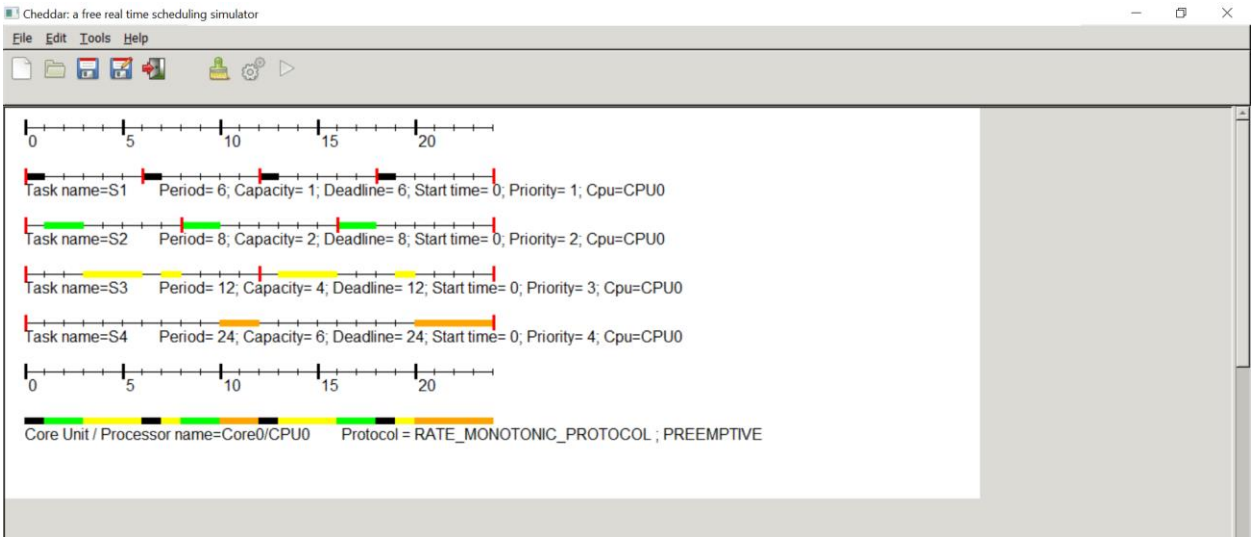
```
Ex-8 U=100.00% (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=6.000000, utility_sum = 0.166667
for 1, wcet=2.000000, period=8.000000, utility_sum = 0.416667
for 2, wcet=4.000000, period=12.000000, utility_sum = 0.750000
for 3, wcet=6.000000, period=24.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.756828
RM LUB INFEASIBLE
```

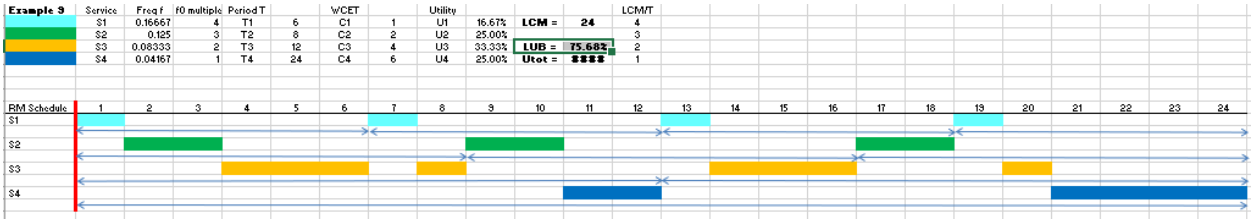
Scheduling Test

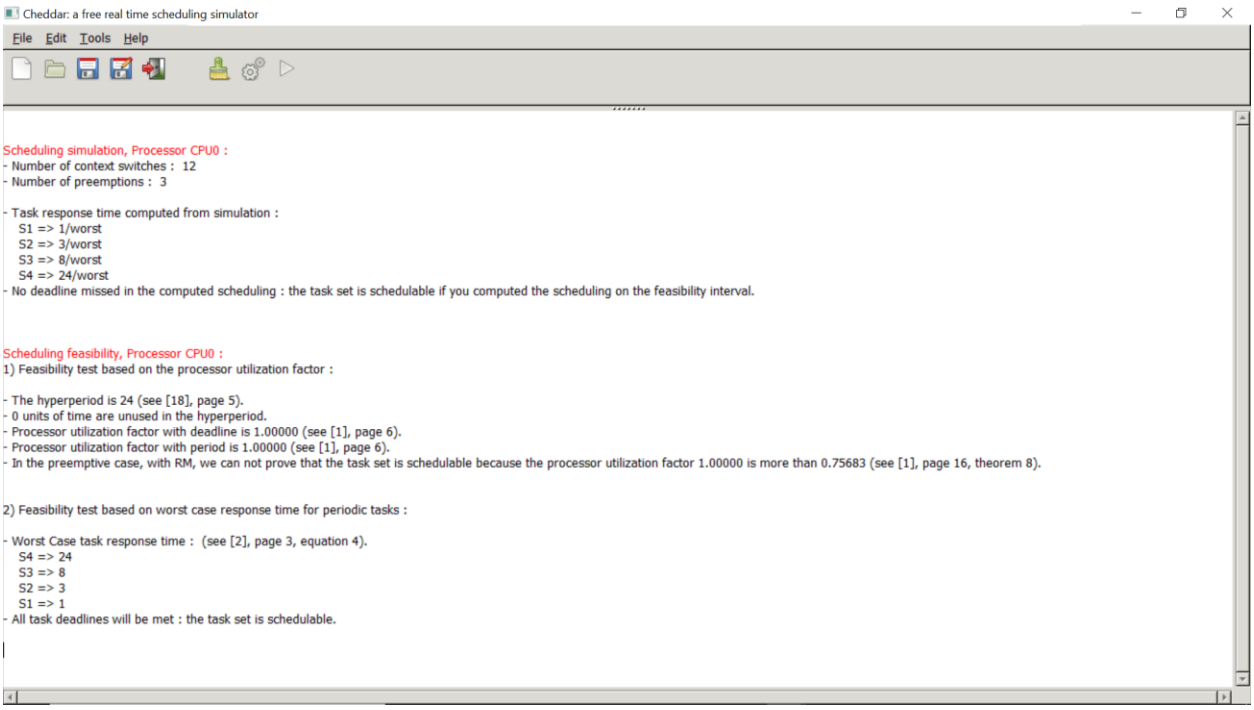
```
Ex-8 U=100.00% (C1=1, C2=2, C3=4, C4=6; T1=6, T2=8, T3=12, T4=24; T=D): FEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=6.000000, utility_sum = 0.166667
for 1, wcet=2.000000, period=8.000000, utility_sum = 0.416667
for 2, wcet=4.000000, period=12.000000, utility_sum = 0.750000
for 3, wcet=6.000000, period=24.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.756828
RM LUB INFEASIBLE
```

Cheddar results

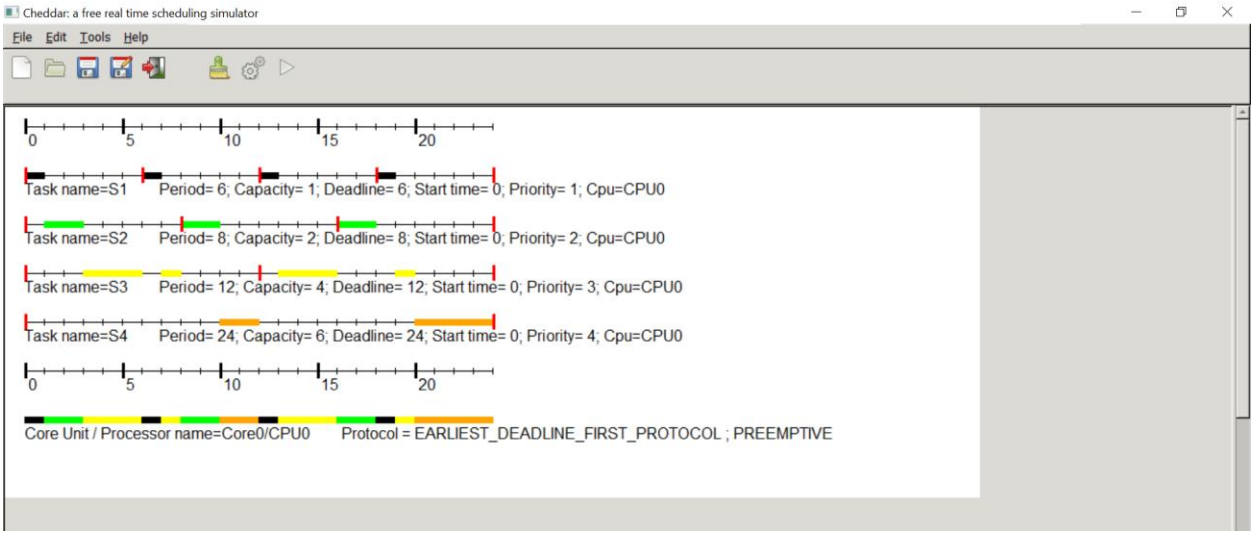
Example 8 RM: Although the Utotal exceeds RMLUB, the scheduling policy is feasible based on the timing diagrams and the cheddar output.

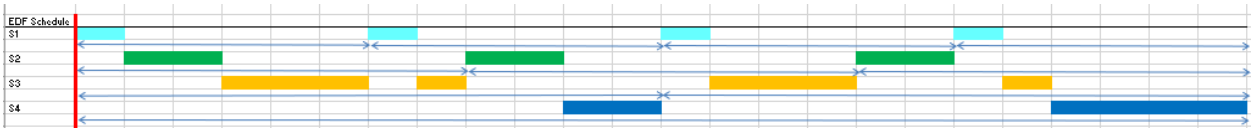


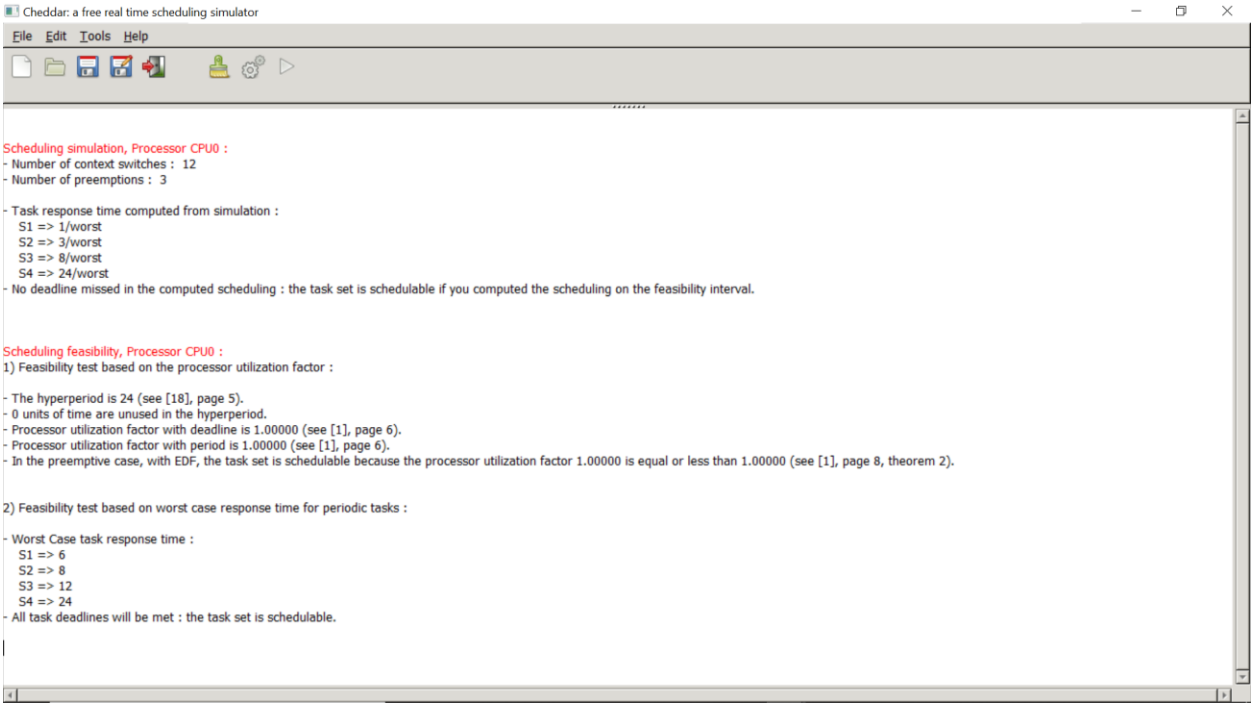




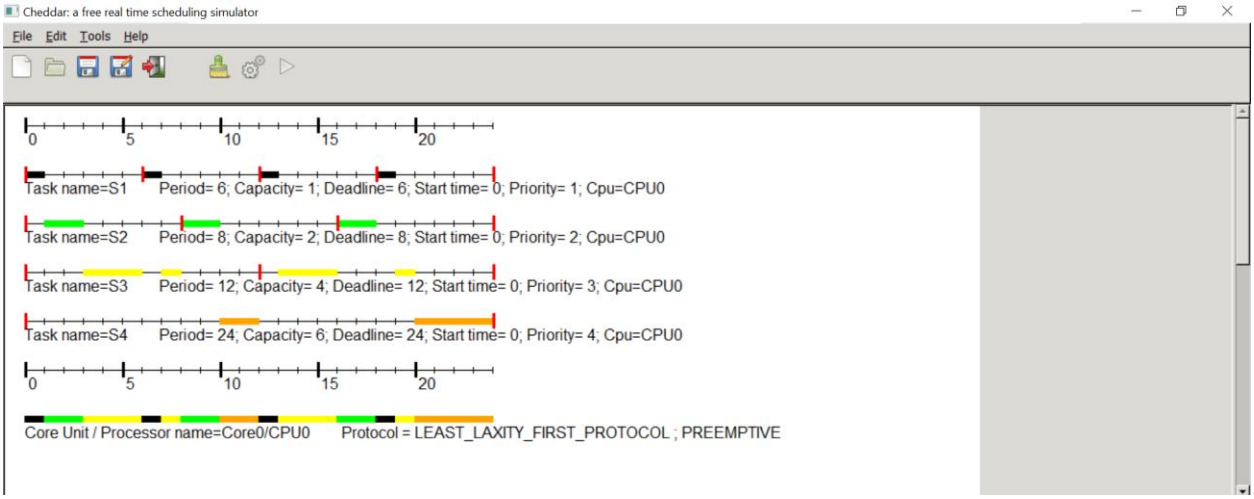
Example 8 EDF: timing diagram in excel matches the cheddar timing diagram. EDF scheduling policy is feasible for given set of services.

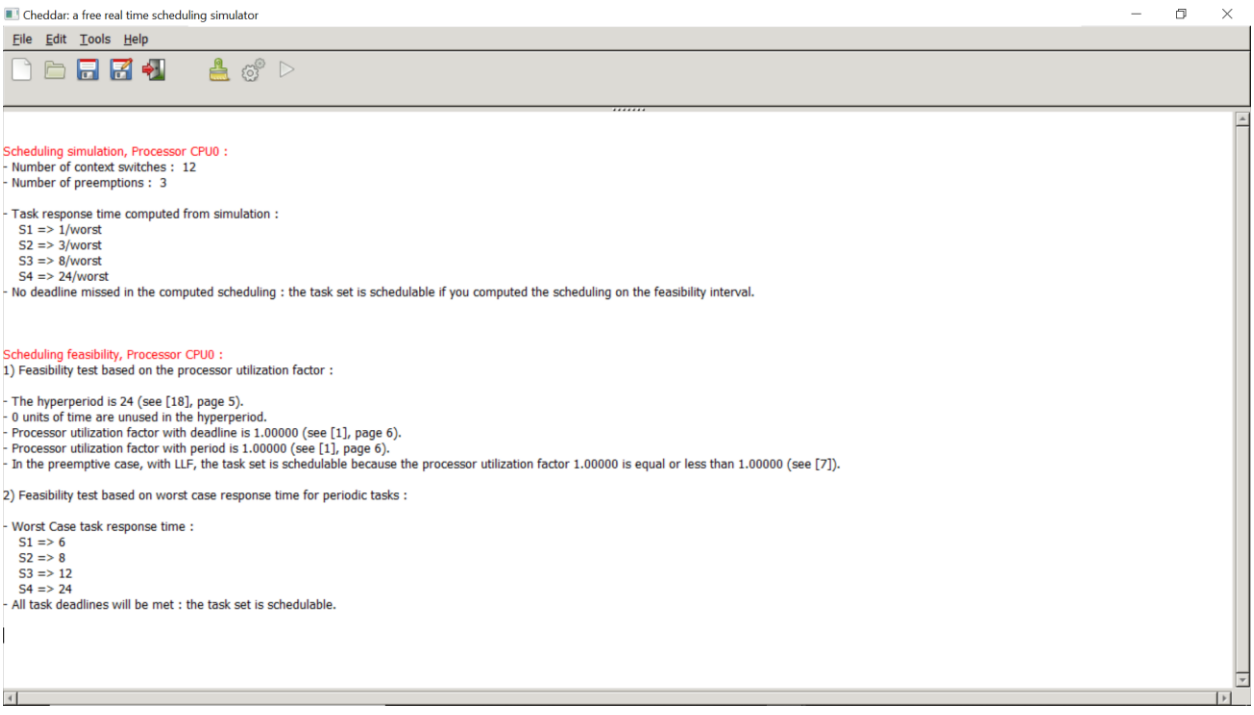






Example 8 LLF: -The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. Although there is difference in the timing diagrams.





SET 9:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1=2	C1=1
S2	T2=5	C2=1
S3	T3=7	C3=1
S4	T4=14	C4=2

The given set of tasks is feasible with RM, EDF and LLF scheduling. The completion and scheduling feasibility point tests confirm the same results. The RMLUB test is not satisfied, however the timing diagrams and cheddar results suggest feasibility for RM Scheduling. EDF and LLF scheduling results and timing diagrams suggest feasible output.

RASPI OUTPUT:

Completion Test

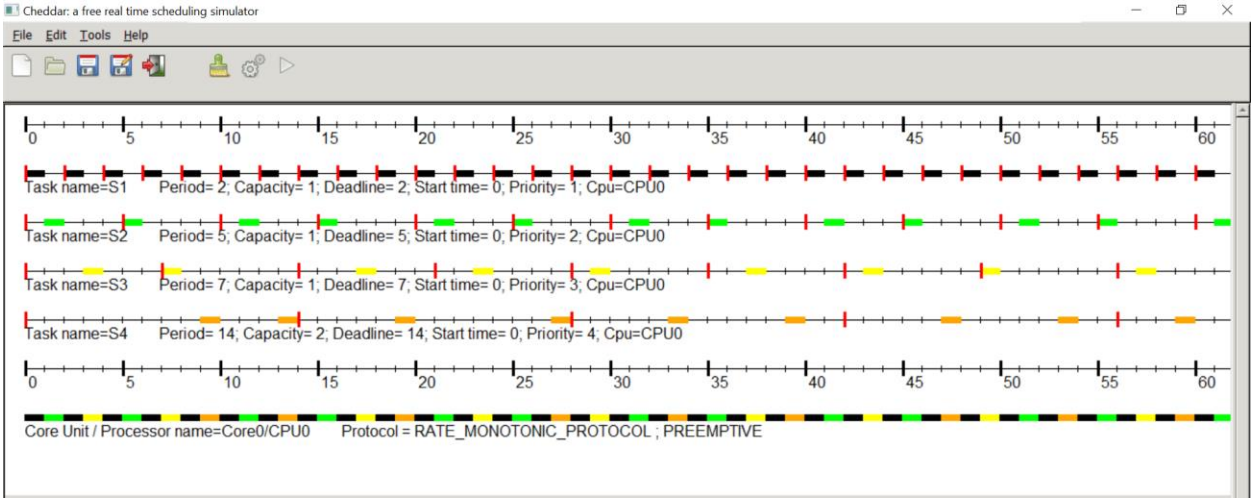
```
Ex-9 U=98.57% (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=1.000000, period=7.000000, utility_sum = 0.842857
for 3, wcet=2.000000, period=14.000000, utility_sum = 0.985714
utility_sum = 0.985714
LUB = 0.756828
RM LUB INFEASIBLE
```

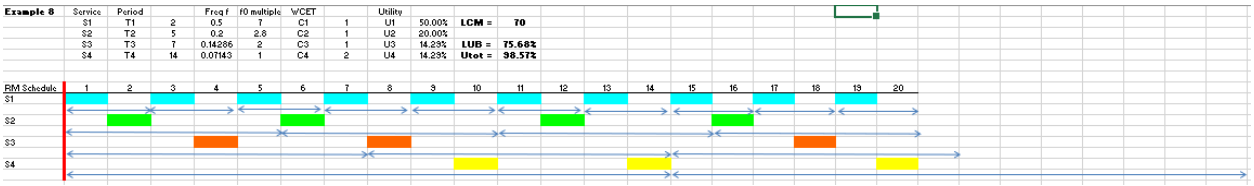
Scheduling Test

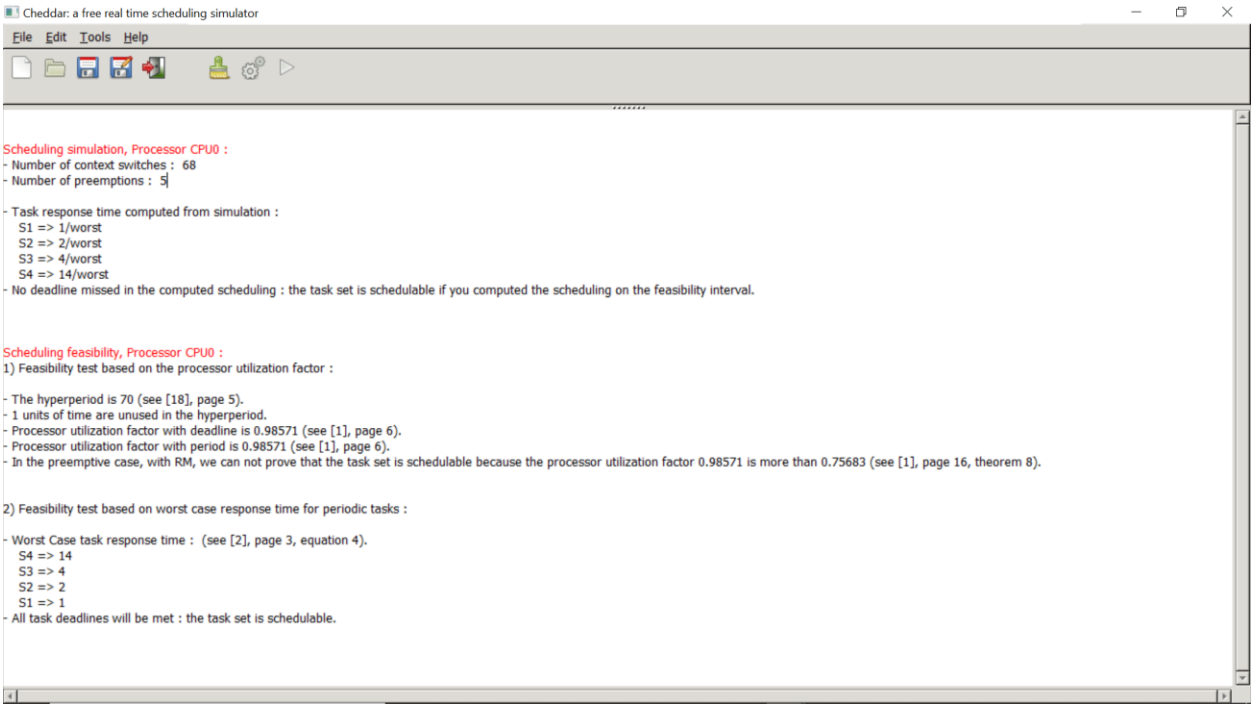
```
Ex-9 U=98.57% (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=14; T=D): FEASIBLE
for 4, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=5.000000, utility_sum = 0.700000
for 2, wcet=1.000000, period=7.000000, utility_sum = 0.842857
for 3, wcet=2.000000, period=14.000000, utility_sum = 0.985714
utility_sum = 0.985714
LUB = 0.756828
RM LUB INFEASIBLE
```

Cheddar results

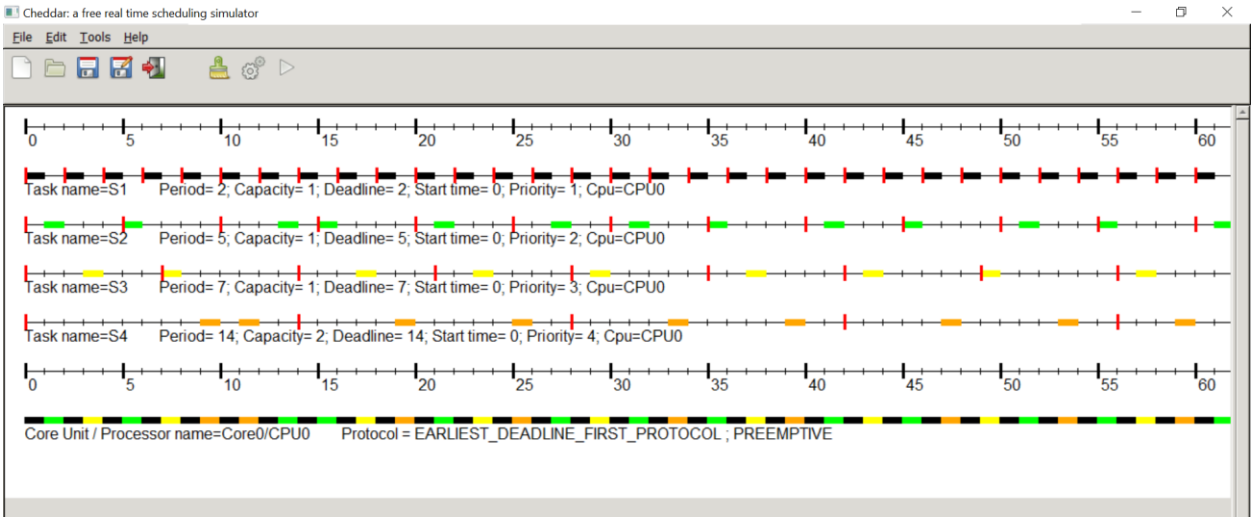
Example 9 RM: Utot =98.57% > RMLUB =75.68% hence there is ambiguity in the safety for this scheduling policy for given set of tasks, although the timing diagrams and cheddar results suggest feasibility.

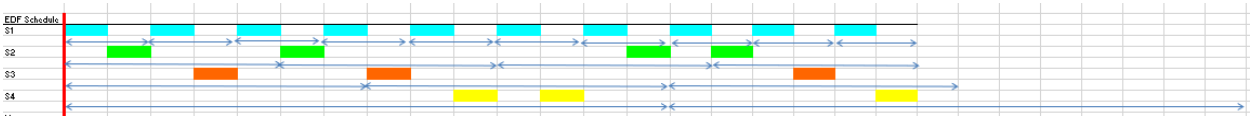


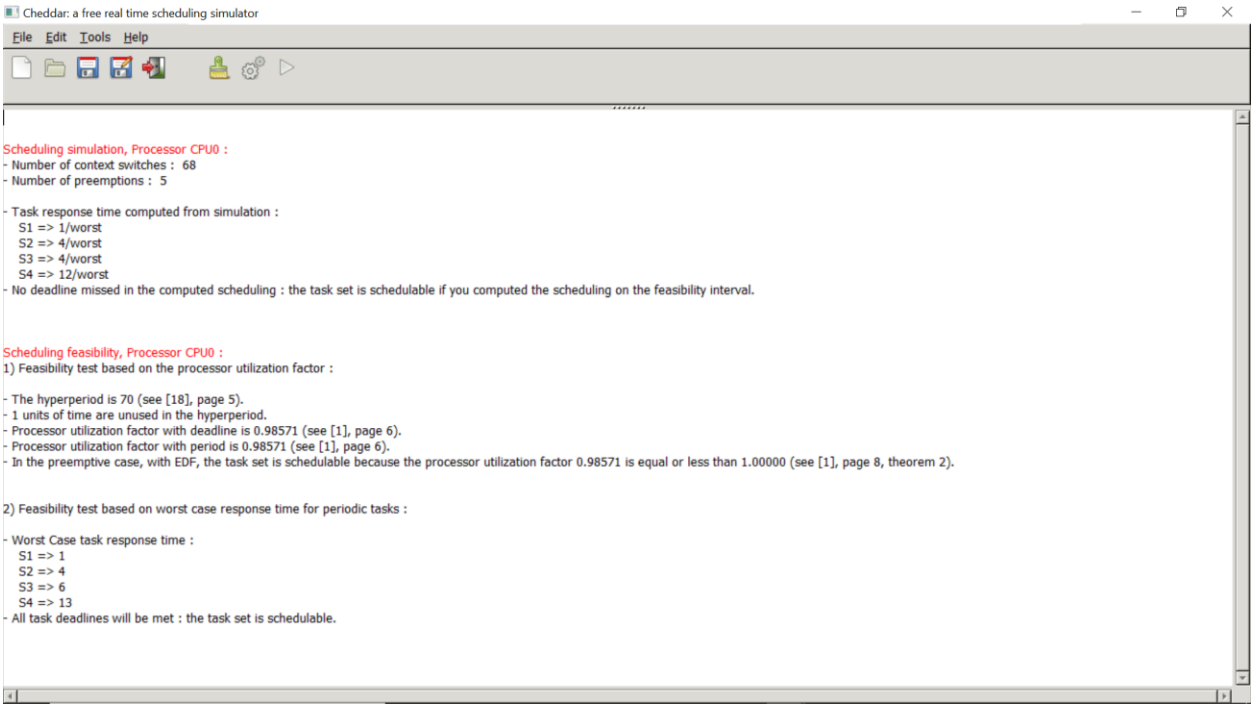




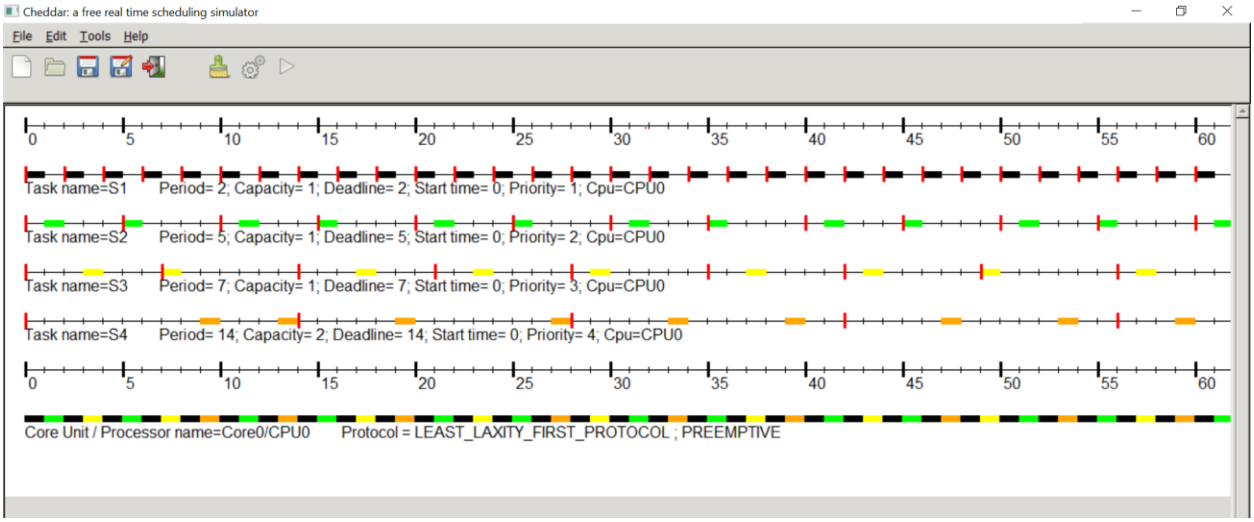
Example 9 EDF: Timing diagrams match for excel sheet and cheddar and the simulation results say that EDF scheduling is feasible for given application.

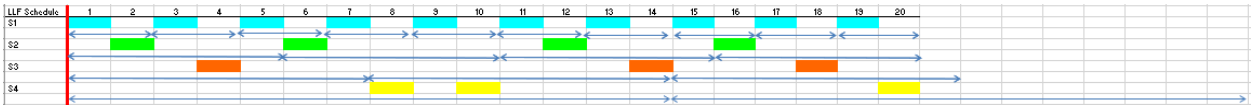


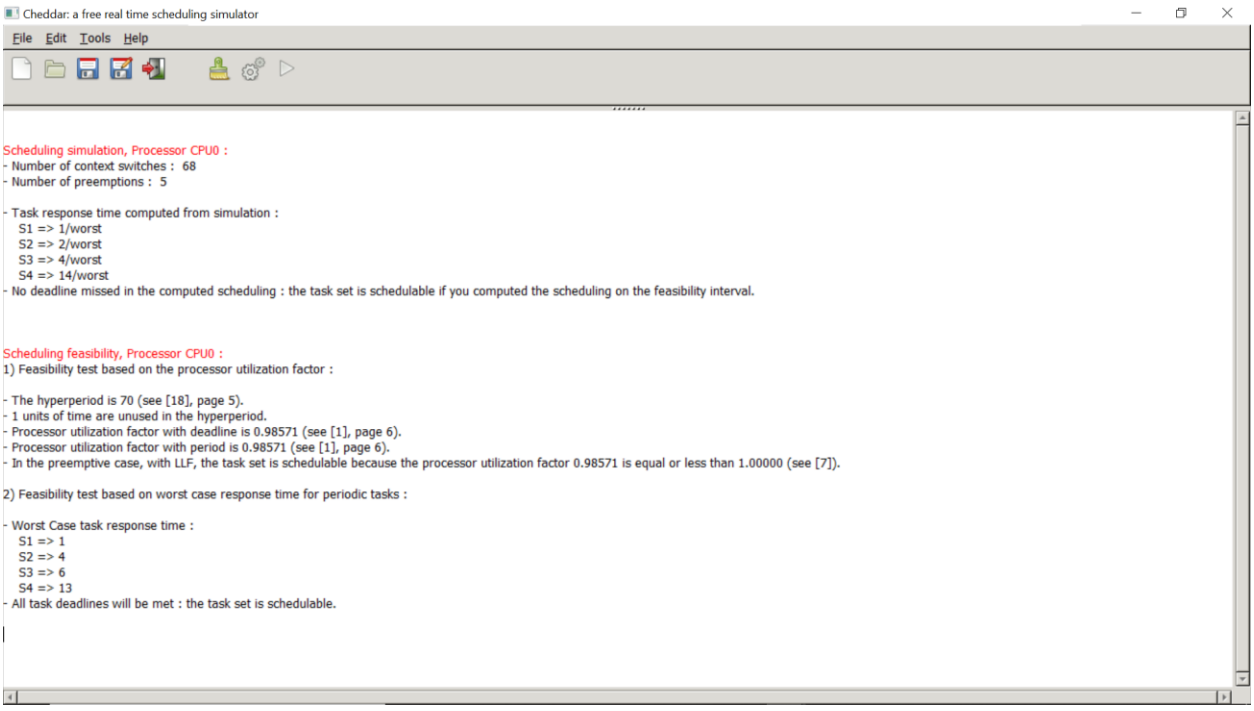




Example 9 LLF: -The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. Although there is difference in the timing diagrams.







SET 10:

S1	T1=3	C1=1
S2	T2=6	C2=2
S3	T3=9	C3=3

RM Scheduling: INFEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

The given set of tasks is infeasible when RM Scheduling is used. It not only fails the RMLUB but also the completion test and scheduling test suggest that it is not feasible to schedule given tasks using RM policy. It can also be asserted using the timing diagrams and cheddar results. On the other hand, EDF and LLF prove to be feasible methods for scheduling given tasks.

RASPI OUTPUT:

Completion Test

```
Ex-10 U=100.00% (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=6.000000, utility_sum = 0.666667
for 2, wcet=3.000000, period=9.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

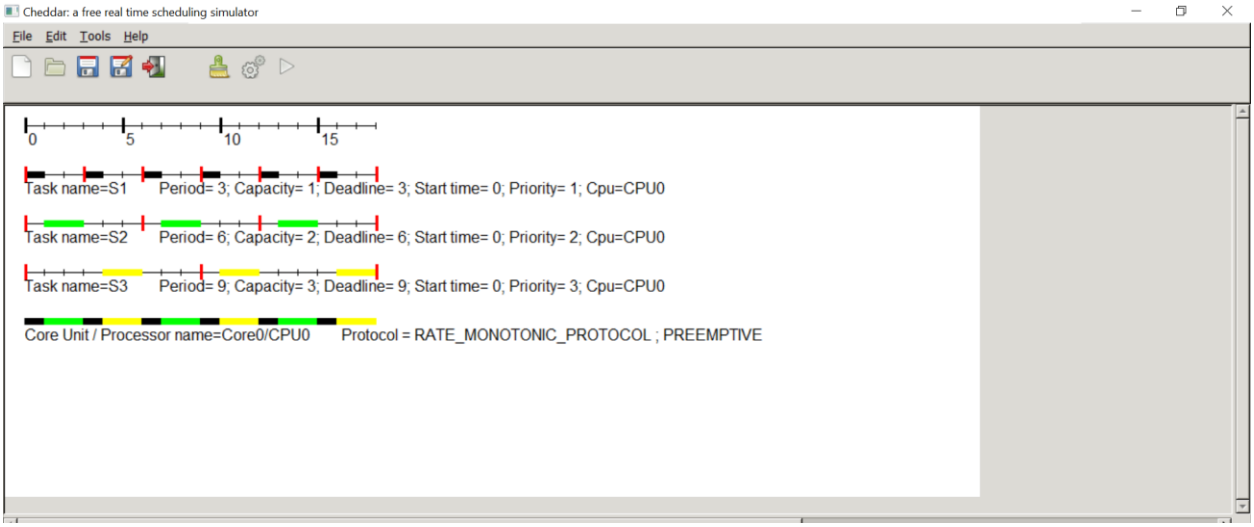
Scheduling Test

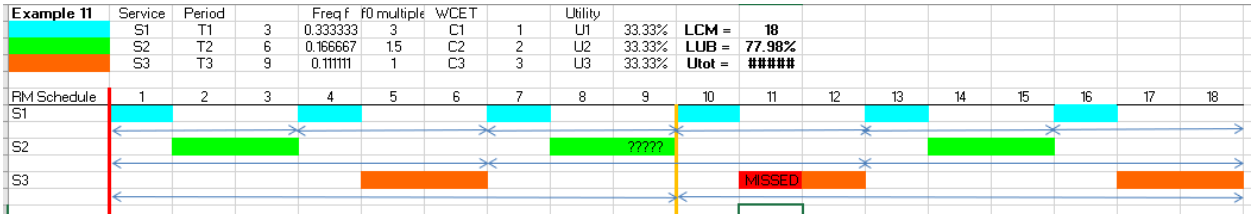
```
Ex-10 U=100.00% (C1=1, C2=2, C3=3; T1=3, T2=6, T3=9; T=D): INFEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=3.000000, utility_sum = 0.333333
for 1, wcet=2.000000, period=6.000000, utility_sum = 0.666667
for 2, wcet=3.000000, period=9.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

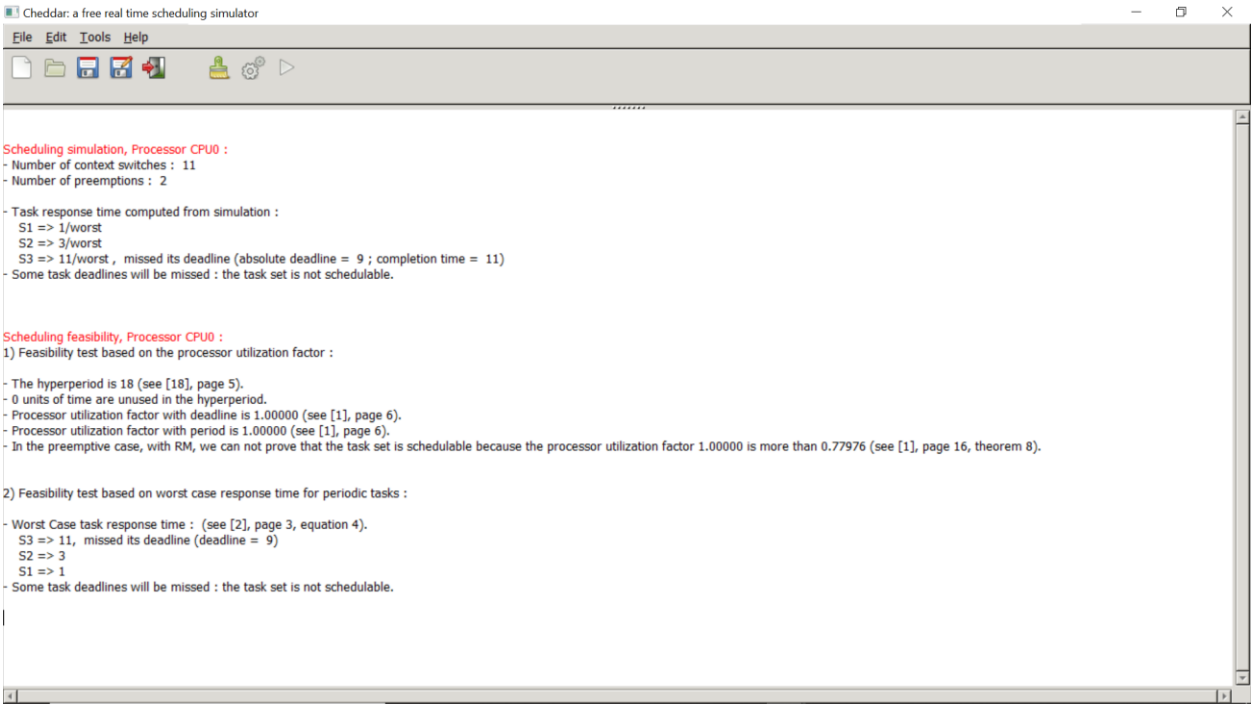
Cheddar results

Example 10 RM: Request for Service 3 is not fulfilled within its deadline and hence RM fails. Also,

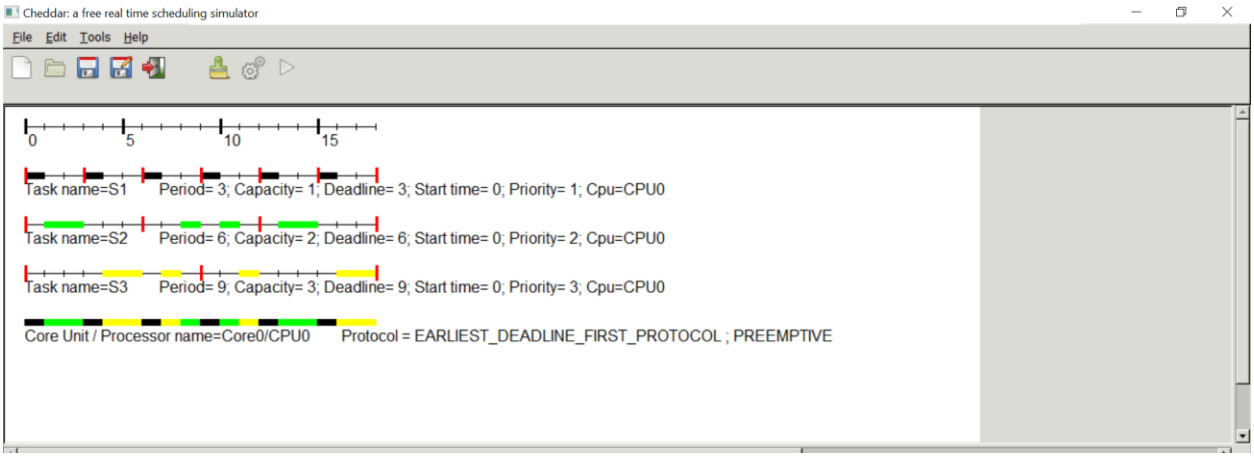
Utot= 100% > RMLUB = 77.9763%

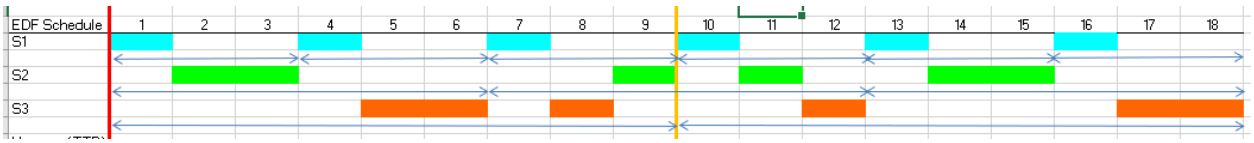


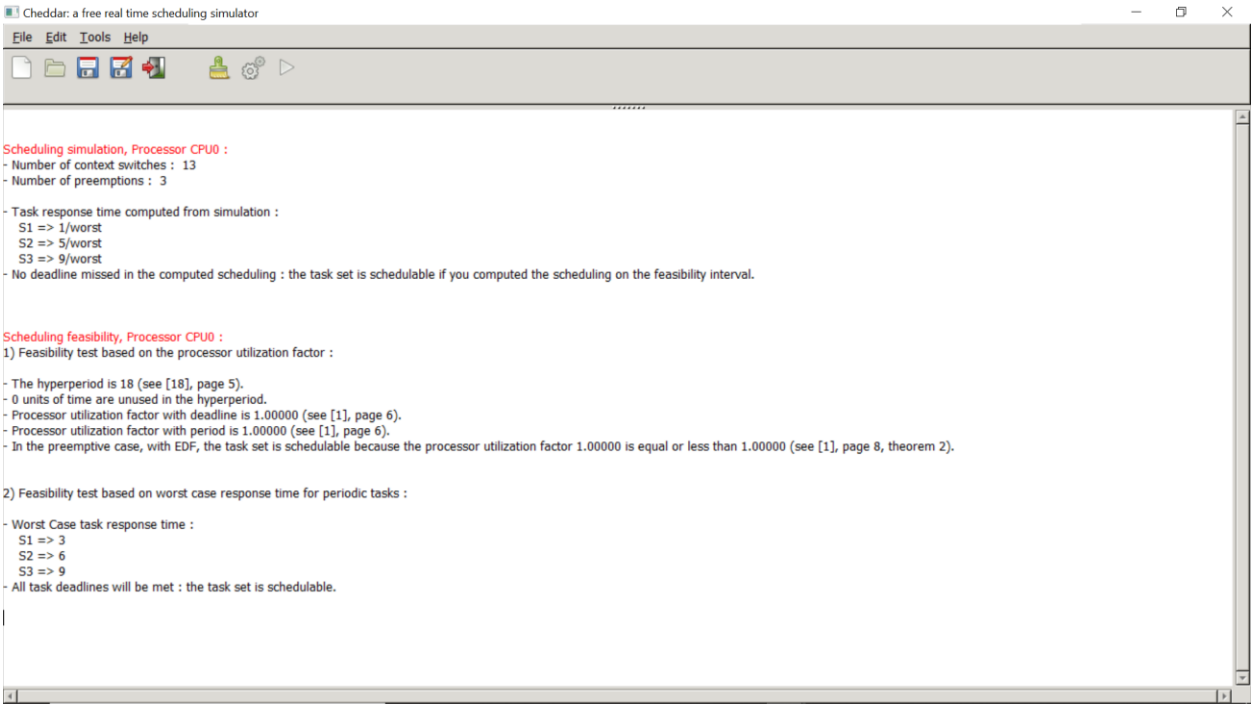




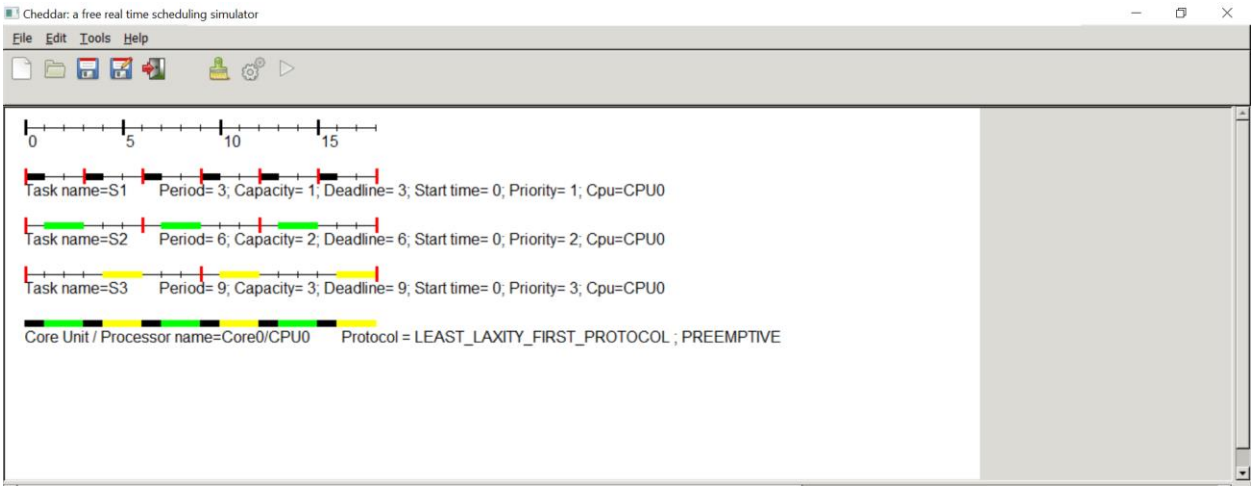
Example 10 EDF: Similar results in excel sheets and cheddar timing diagrams. Simulation and feasibility results prove EDF to be a feasible policy for given the given set.

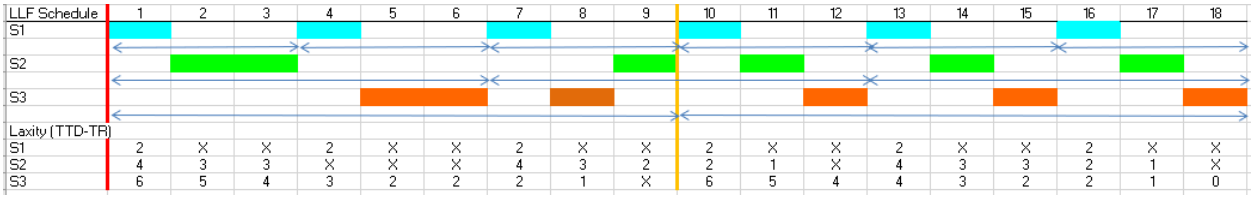


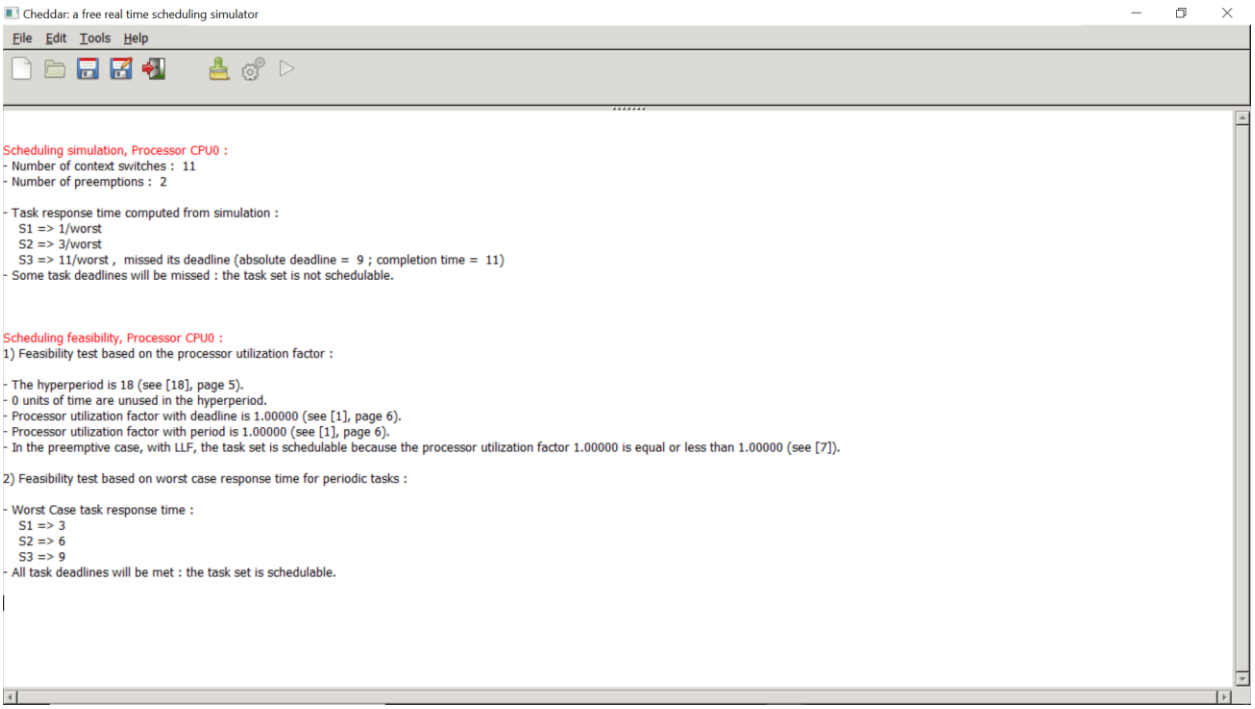




Example 10 LLF: The task set can be scheduled using LLF as seen in the timing diagram and as described in the below screenshot in the scheduling feasibility tests run by Cheddar. Although there is difference in the timing diagrams.







SET 11:

RM Scheduling: FEASIBLE
EDF Scheduling: FEASIBLE
LLF Scheduling: FEASIBLE

S1	T1=2	C1=1
S2	T2=4	C2=1
S3	T3=8	C3=2

Fails the RM LUB test but feasible results using cheddar and timing diagrams for RM. EDF and LLF provide feasible scheduling. Completion tests and Scheduling point feasibility tests also confirm RM scheduling to be feasible.

RASPI OUTPUT:

Completion Test

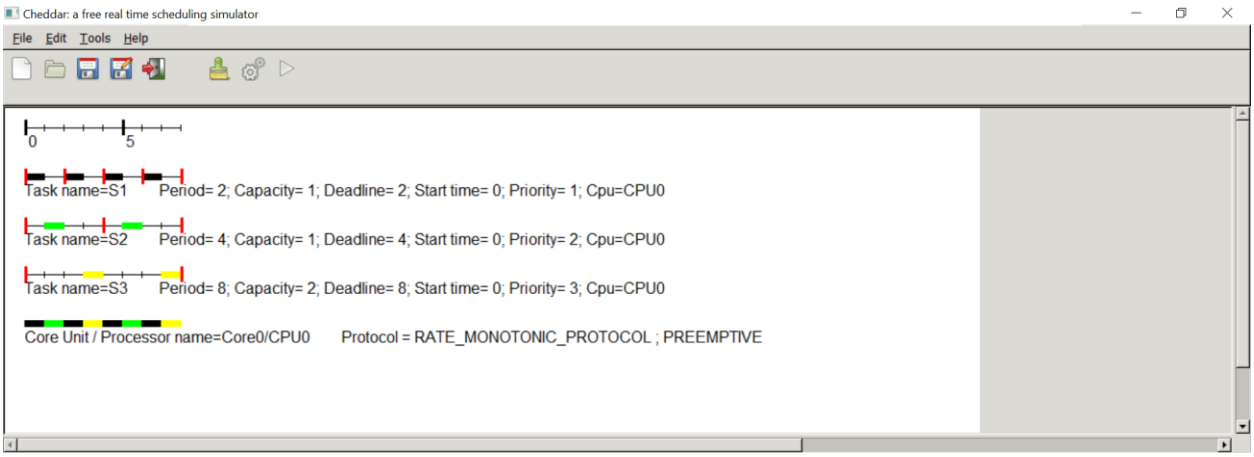
```
Ex-11 U=100.00% (C1=1, C2=1, C3=2; T1=2, T2=4, T3=8; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=2.000000, period=8.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

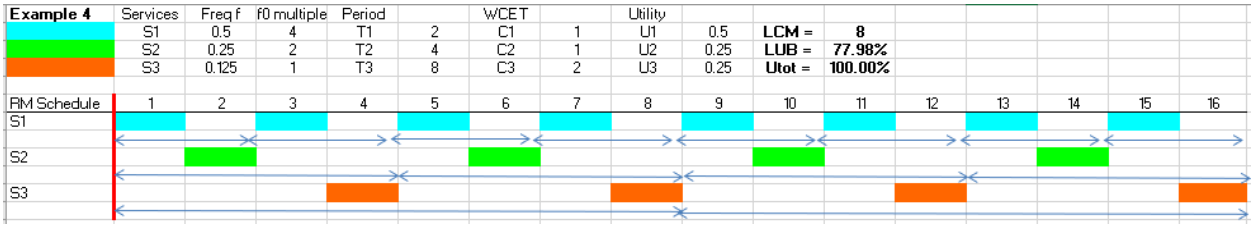
Scheduling Test

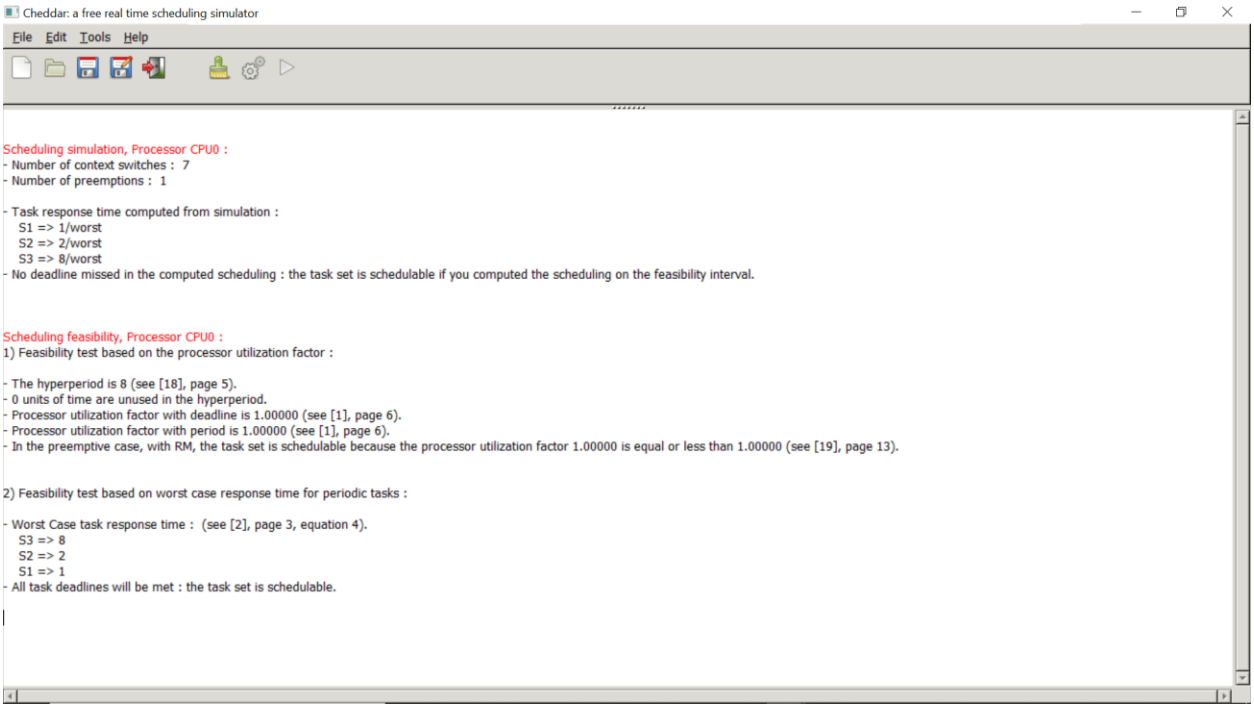
```
Ex-11 U=100.00% (C1=1, C2=1, C3=2; T1=2, T2=4, T3=8; T=D): FEASIBLE
for 3, utility_sum = 0.000000
for 0, wcet=1.000000, period=2.000000, utility_sum = 0.500000
for 1, wcet=1.000000, period=4.000000, utility_sum = 0.750000
for 2, wcet=2.000000, period=8.000000, utility_sum = 1.000000
utility_sum = 1.000000
LUB = 0.779763
RM LUB INFEASIBLE
```

Cheddar results

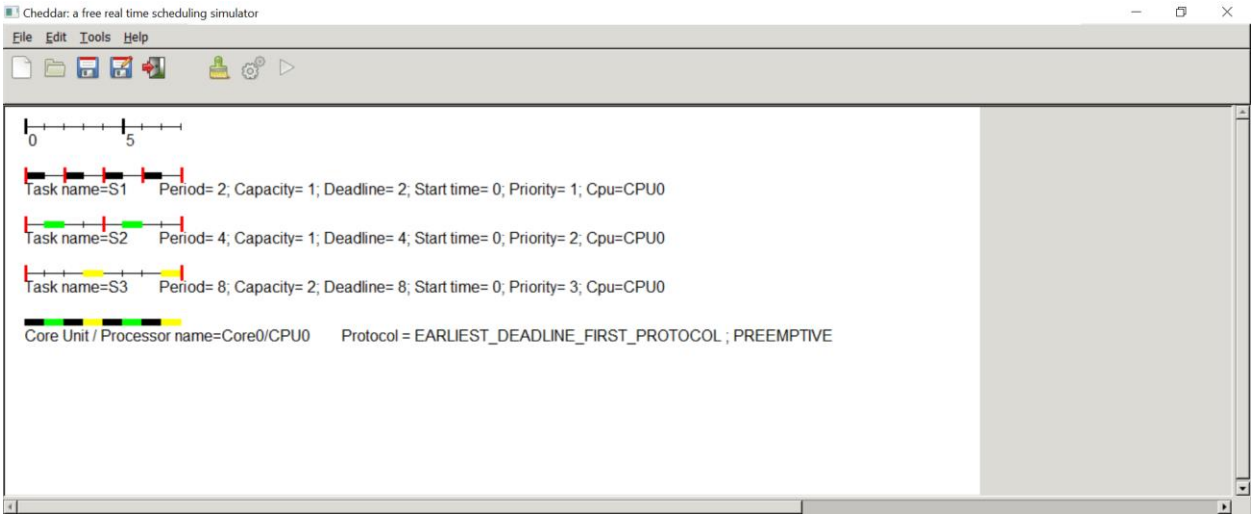
Example 11 RM: not feasible using RMLUB test but the code output for completion and scheduling tests confirm feasibility. It can also be observed using timing diagrams and cheddar results

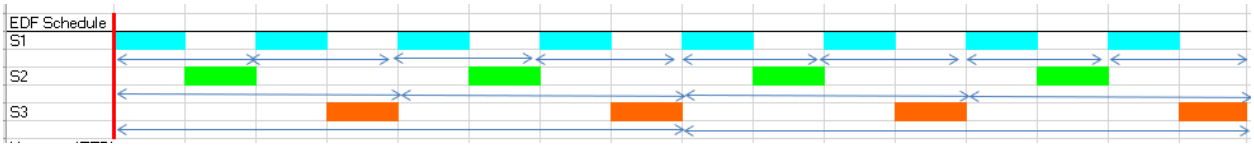


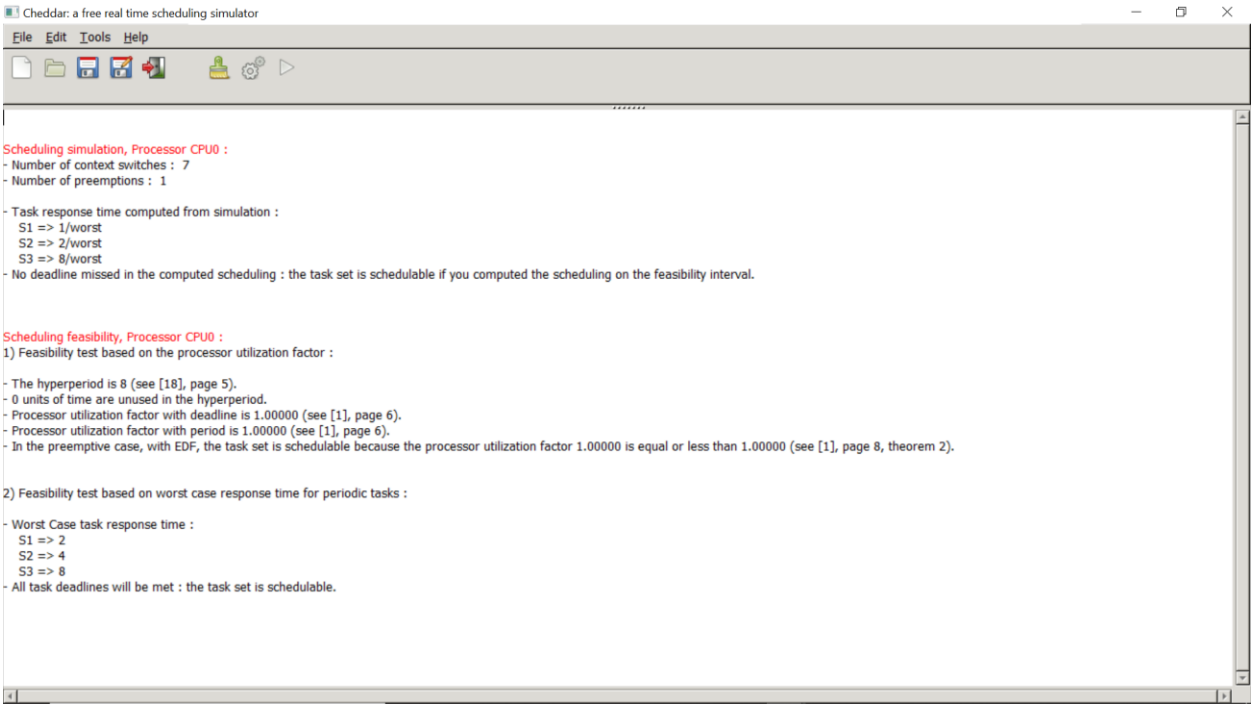




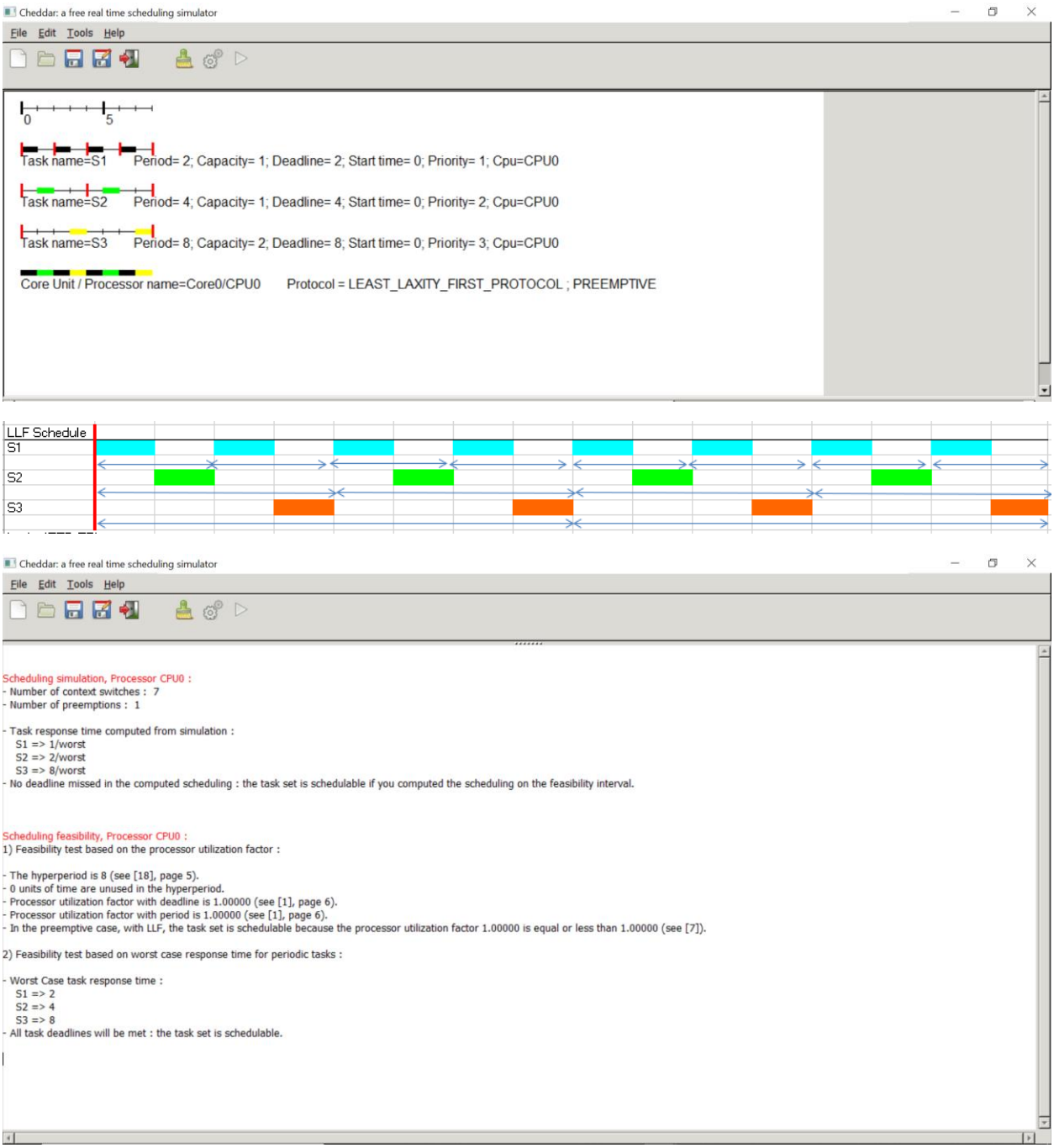
Example 11 EDF: Similar results for Cheddar as in the excel sheet timing diagrams. Feasible schedule with EDF. (The cheddar timing diagram is just provided until the LCM, the excel timing diagram is repeated after the LCM.)







Example 11 LLF: Timing diagram for scheduling policy matches in excel and cheddar. The simulation results below confirm the same.



c) Does your modified Feasibility code agree with Cheddar analysis in all 5 additional cases? Why or why not?

For the additional cases in the code, I have tried to add all cases including change in number of services, harmonic set of services, overload case, sets that cannot be scheduled using RM and hence the analysis of EDF and LLF is also provided.

```
// U=1.06
U32_T ex5_period[] = {2, 4, 16};
U32_T ex5_wcet[] = {1, 1, 5};

// U=1.0
U32_T ex6_period[] = {2, 5, 10};
U32_T ex6_wcet[] = {1, 2, 1};

// U=1.0
U32_T ex7_period[] = {3, 5, 15};
U32_T ex7_wcet[] = {1, 2, 4};

// U=1.0
U32_T ex8_period[] = {6, 8, 12, 24};
U32_T ex8_wcet[] = {1, 2, 4, 6};

// U=0.9847
U32_T ex9_period[] = {2, 5, 7, 14};
U32_T ex9_wcet[] = {1, 1, 1, 2};

// U=1.0
U32_T ex10_period[] = {3, 6, 9};
U32_T ex10_wcet[] = {1, 2, 3};

// U=1.0
U32_T ex11_period[] = {2, 4, 8};
U32_T ex11_wcet[] = {1, 1, 2};
```

I have added the completion tests and the scheduling point feasibility tests for above given set of services in the existing code.

The results provided for the rate monotonic scheduling policy match for all the cases. The completion and the scheduling point feasibility test results confirm if the given set of tasks are feasible or not and these results match the cheddar simulation and the feasibility check outputs. There are cases where the set if services do not pass the RMLUB tests, yet the timing diagrams and the tests suggest feasibility. Cheddar outputs affirm this as well and the result is exactly same in the case of RM.

The results for EDF (Earliest deadline first) match in the cheddar tool and the excel sheet diagrams for all the additional cases. There are no feasibility tests in the code for EDF, hence the analysis is performed based on given two methods.

The results for LLF (Least Laxity first) are ambiguous. The Scheduling simulation results show the worst-case analysis for the given task and suggest that the scheduling method is not feasible. While, the scheduling feasibility results suggest feasibility using LLF algorithm. Another peculiar result that I observed in the output was difference in the timing diagrams. There are some examples where I have mentioned in the description if there are differences in the timing diagrams given in excel an the one generated by cheddar tool. The timing diagrams in excel are verified and in my opinion provide a clear inspection of the policy based on the given specifications if it would be feasible or not. Although, the results on cheddar are ambiguous (just in case of LLF).

Conclusion: The results concerning the feasibility of any scheduling policy, the timing diagrams, code tests and the cheddar output match but there are variations in terms of timing diagrams (ambiguity is just in case of LLF).

Question 4) Read Chapter 3 of the textbook.
a) Provide 3 constraints that are made in 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.

Constraints made in the RM LUB derivation:

• C1: Deadline = Period by Definition
For simplicity in calculations, request period is considered equal to the deadline for a given task. This does not consider cases where the deadline is before the request period or sometimes it may exceed(overload).

• C2: Fixed Priority, Preemptive, Run-to-Completion Scheduling
The policies taken in consideration for deriving important conclusions and results for a specific set of tasks are considered to be fixed priority, preemptive, run-to-completion. The dynamic priority allocation, asynchronous interrupts are hardly talked about.

• C3: Only CPU Utilization taken into consideration.
The algorithmic calculations and derivations are provided only for the CPU utilization. Other resources like shared memory, IO, interfaces are not taken into account.

• C4: Critical Instant, critical time zone.
Calculations done based on the worst-case execution time. Critical instant is the time when a task would take the longest to respond when all the services are requested simultaneously. (lowest priority task preempted by all other tasks).

Assumptions made in Liu and Layland paper:

• A1: All Services Requested on Periodic Basis, the Period is Constant
Statement: *"The requests for all tasks for which hard deadlines exist are periodic, with constant interval between requests."*
This assumption expects all the incoming tasks having a hard deadline are periodic i.e. they are requested at a regular interval of time. The time in between each request period is constant for each interval.

• A2: Completion-Time < Period
Statement: *"Deadlines consist of run-ability constraints only-i.e., each task must be completed before the next request for it occurs."*

It is one of the constraints (as explained earlier), the deadline is considered equal to the request period of the service. Deadlines define the time up-to which the CPU is expected to complete the task for normal operation of the system (no overloading). The current statement assumes the job scheduled for the CPU to be serviced is completed before next request for the task arrives. Hence, the requests do not coincide, and they can be served before the deadline. If the system behaves in such a manner the queuing problems can be eliminated easily for individual tasks.

• A3: Service Requests are Independent (No Known Phasing)
Statement: *"The tasks are independent in that requests for a certain task do not depend on the initialization or the completion of requests for other tasks."*
This can be better explained with an example. If a certain task has request period 2 and another task has request period 9, there is a possibility that at a certain instant, both the tasks request service at the same time. This may cause the scheduler pick one depending on the priority and the other may or may not miss the deadline. This phenomenon is called phasing. Liu and Layland do not take this phenomenon into account.

• A4: Run-time is Known and Deterministic (WCET may be Used)
Statement: *"Run-time for each task is constant for that task and does not vary with time. Run-time refers to the time which is taken by a processor to execute the task without interruption."*
The three parameters we require to determine the feasibility and safety of the scheduling algorithm are the run-time of the service, request period/ request rate and the deadline for the respective service. Liu ad Layland have considered the deadline and the run-time to be equal or neglected the deadline and considered the worst-case execution time (even with worst case time, the service is processed before the deadline) for simplicity in calculations. If we consider the run-time of each task to be constant and periodic (does not vary with time), it is easier to schedule the services for a particular period.

b) 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.

- Theorem 3 in the Liu and Layland proves the RMLUB equation for a set of two tasks and based on these results we can derive for any number of tasks.
  For a set of two tasks with request periods *T1* and *T2*, run times *C1* and *C2* respectively. Assuming the request period for 2 is greater than 1, i.e. service 1 is requested more frequently. Hence, according to RM policy, priority of task 1 is greater than that of T2.
- For a period of T2, there would be ⌈T2/T1⌉ requests of T1.
- We consider two cases where for the first one the run time for task one is within the critical time zone and for the second one the run time for task two exceeds the critical time zone.
  Case 1. The run-time C1 is short enough that all requests for service 1 within the critical time zone of T2 are completed before the second request for service 2. That is,

$$C_1 \leq T_2 - T_1\lfloor T_2/T_1 \rfloor.$$

Hence, the time remaining would be available for serving the request for second one.

$$C_2 = T_2 - C_1\lceil T_2/T_1 \rceil.$$

Thus, the utilization factor for this case can be given as,

$$U = 1 + C_1[(1/T_1) - (1/T_2)\lceil T_2/T_1 \rceil].$$

U is monotonically decreasing with increase in C1.

Case 2. The execution of the ⌈T2/T1⌉ request for service 1 overlaps the second request for service 2. In this case,

$$C_1 \geq T_2 - T_1\lfloor T_2/T_1 \rfloor.$$

C2, can be derived based on the number of occurrences of service request for 1 and the time taken by it.

$$C_2 = -C_1\lfloor T_2/T_1 \rfloor + T_1\lfloor T_2/T_1 \rfloor$$

- Depending on these equations of C1 and C2, U can be calculated as,

$$U = (T_2/T_1)\lfloor T_2/T_1 \rfloor + C_1[(1/T_1) - (1/T_2)\lfloor T_2/T_1 \rfloor].$$

U is monotonically decreasing with increase in C1.

- It was difficult to understand the utilization factor derivation when run time C1 is exactly equal to the critical time zone.

$$C_1 = T_2 - T_1\lfloor T_2/T_1 \rfloor$$

- As both the equations for C2 are valid, considering second one, we obtain,

$$U = 1 - (T_1/T_2)[\lceil T_2/T_1 \rceil - (T_2/T_1)][(T_2/T_1) - \lfloor T_2/T_1 \rfloor].$$

- To simplify the equation we consider,

$$I = \lfloor T_2/T_1 \rfloor \text{ and } f = \{T_2/T_1\}.$$

I is the integer value for number of times T1 occurs during period T2.
f is the fractional time of the last release of T1 during T2, lowest value of f=0.

We obtain U as,

$$U = 1 - \left(\frac{f(1-f)}{(I+f)} \right)$$

Given equation is derived considering the floor is equal to ceiling when f is non zero.
Minimum U occurs at smallest possible value of I which is 1, hence U can be written as

$$U = 1 - \left(\frac{(f - f^2)}{(1+f)} \right)$$

Finding first derivative of U, and equating it to zero, we get

$$dU/df = \frac{(1+f)(1-2f)-(f-f^2)(1)}{(1+f)^2} = 0$$

Which gives a value of f = ($2^{1/2}$ – 1).

And hence, U= 2($2^{1/2}$ – 1), which is the equation for RMLUB for two services and can be similarly derived for any number of services.

Hence RM LUB,

$$U = m(2^{1/m} - 1),$$

- Processor utilization factor to be the fraction of processor time spent in the execution of the task set. In other words, the utilization factor is equal to one minus the fraction of idle processor time. Since Ci/Ti is the fraction of processor time spent in executing task Ti for m tasks, the utilization factor is:

$$U = \sum_{i=1}^{m} (C_i/T_i)$$

- The total utilization factor should be less than the RMLUB for the schedule to be feasible. It is a simple and quick feasibility check that is sufficient.
- Although it is not a necessary test which means we can safely utilize a CPU at levels below 100% but above the RM LUB and prove the schedule as feasible. (supported by the Lehoczky, Sha, Ding theorem).
- Completion test and the Scheduling point test are algorithm for Necessary and Sufficient feasibility testing with RM policy.

Scheduling Point Test

The Lehoczky, Sha and Ding theorem states that "*if a set of services can be shown to meet all deadlines from the critical instant up to the longest deadline of all tasks in the set, then the set is feasible*."

Based on the assumptions made in Liu and Layland paper, Lehoczky, Sha, and Ding introduced an iterative test for this statement as:

$$\forall i, 1 \le i \le n, \min \sum_{j=1}^{i} C_j \left\lceil \frac{(l)T_k}{T_j} \right\rceil \le (l)T_k$$

$$(k,l) \in R_i$$

$$R_i = \left\{ (k,l) \middle| 1 \le k \le i, l = 1, \ldots, \left\lfloor \frac{T_i}{T_k} \right\rfloor \right\}$$

- Where n is the number of tasks in the set Si to Sn, where S1 has higher priority than S2, and
- Sn has higher priority than S(n>1).
- j identifies Sj, a service in the set between S1 and Sn.
- k identifies Sk, a service whose l periods must be analyzed.
- l represents the number of periods of Sk to be analyzed.
- $\left\lceil \frac{(l)T_k}{T_j} \right\rceil$ represents the number of times Sj executes within l period of Sk.
- $C_j \left\lceil \frac{(l)T_k}{T_j} \right\rceil$ is the time required by Sj to execute within l periods of Sk—if the sum of these times for the set of tasks is smaller than l period of Sk, then the service set is feasible.

Completion time test

The Completion Time Test is presented as an alternative to the Scheduling Point Test:

$$a_n(t) = \sum_{j=1}^{n} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

- $\left\lceil \frac{t}{T_j} \right\rceil$ is the number of executions of Sj at time t.
- $\left\lceil \frac{t}{T_j} \right\rceil C_j$ is the demand of Sj in time at t.
- $a_n(t)$ is the total cumulative demand from the n tasks up to time t.

Passing this test requires proving that an(t) is less than or equal to the deadline for Sn, which proves that Sn is feasible. Proving this same property for all S from S1 to Sn proves that the service set is feasible.

References

a) REAL-TIME EMBEDDED COMPONENTS AND SYSTEMS with LINUX and RTOS by *Sam Siewert & John Pratt.*

b) Architecture of the Space Shuttle Primary Avionics Software System by *Gene D. Carlow.*

c) Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment by *C. L. Liu* (Project MAC, Massachusetts Institute of Technology) & *James W. Layland* (Jet Propulsion Laboratory, California Institute of Technology)