NUS
National University of Singapore

Faculty of Engineering

Module	EE5903
Title	Evaluating real time system algorithms (EDF and RMS) with its enhanced algorithm (EFDF and RMZL)
Name	Zavier Ong Jin Jie
Matriculation	A0138993L
Mobile Number	92223270

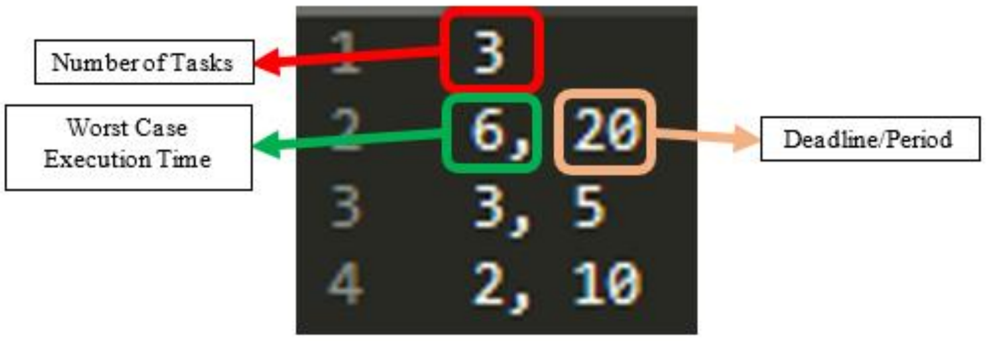
Abstract
In this paper, two well-known algorithms: (1) Earliest Deadline First (EDF), (2) Rate Monotonic Scheduling (RMS) as well as its enhanced versions: (1) Earliest Feasible Deadline First (EFDF), (2) Rate Monotonic-Zero Laxity (RMZL) will be implemented in Python 3. The aim of this paper is to compare the performance of the original algorithms with its enhanced versions as well as evaluate the strengths and weaknesses of these algorithms.

This paper will experiment with several task sets for each class of algorithms. The result of the evaluation and comparison depends heavily on the purpose of the real time system and how the algorithms handle tasks in similar zero laxity conditions.

Introduction
There are two types of real time systems: (1) Soft real time systems and (2) Hard real time systems. Soft real time is a property of the timeliness of a computation where the value diminishes according to its lateness. A soft real time system can accept some delays in the task completion if there is still some value to completing the task [2]. A hard real time system, however, is when there is no value to a computation if the task exceeds its deadline and the effects of a late computation may be catastrophic to the system [2]. Therefore, it is imperative that these tasks need to follow a scheduling algorithm to ensure the deadlines for the tasks assigned to be met.

In this paper, the two main algorithms to be evaluated would be the EFDF and the RMZL. Since the EFDF is a dynamic priority scheduling class while the RMZL in its essence is a static-priority scheduling class, it would be unfair to compare them together. Therefore, the EFDF would be evaluated and compared against its original algorithm, the EDF while the RMZL would be evaluated and compared against RMS.

Description of Input Data
The input data for all the task sets are generated by hand in the form of a text file.



```
3
6, 20
3, 5
2, 10
```

Figure 1. Sample of Task set text file

As shown in Figure 1 above, the first line represents the number of tasks the task set contains. All subsequent lines in the text file are the details of each task, where each line represent one task. The first parameter is the worst-case execution time (WCET), and the second parameter would be the deadline and period. This research paper will follow the assumptions made in accordance with the scholarly articles studied this paper:

1. All tasks are independent from each other.
2. There is no task that require resources from another task.
3. All tasks have their deadline equivalent to their periods.
4. All tasks do not have its own priority. (Priority is set entirely based on the algorithm)
5. When a task has reached zero laxity, the task will have its priority raised to the highest and no other task will be able to preempt it.
6. When a task has reached negative laxity, the task will not be scheduled.

For all task sets, the hyper period of the task set is calculated, and the simulation would be executed for duration equivalent to the hyper period. This is because there is no need to run the simulation more than the hyper period as it would just replicate the same schedule scheduled in the first hyper period.

As mentioned in the abstract, there will be two different sets of tasks sets:

EDF & EFDF

For the EDF and EFDF task sets, though the tasks sets would be different from the RMS and RMZL task sets, the task sets that are tested on EDF would be tested on EFDF as well. For this simulation, a total of three different tasks sets are prepared:

1. Small task set where utilization bound <= 1.
2. Small task set where utilization bound > 1.
3. Large task set where utilization bound > 1 (Similar to RMS large task set).

As the EDF algorithms rely on the WCET to determine a task priority, the tasks set WCET parameter would be varied more as compared to the deadline/period.

RMS & RMZL

Like EDF and EFDF, there will be three different task sets prepared for this simulation.

1. Small task set where utilization bound <= 1.
2. Small task set where utilization bound > 1.
3. Large task set where utilization bound > 1 (Similar to EDF large task set).

Unlike EDF and EFDF, since the RMS algorithm determines its priority on the period of the task, the task set would differ from EDF's task set on the deadline/period parameter instead of the WCET parameter in the task set text file.

Algorithm Description

EDF

When implementing the EDF algorithm, it is imperative to first calculate the utilization bound of the task set to know if the task set is schedulable in the first place. The calculation of the utilization bound, also known as the schedulability test, follows this equation:

$$U = \sum_{i=1}^{n} C_i / T_i \leq 1$$

C_i → worst-case computation times of the n processes

T_i → inter-arrival periods (assumed to be equal to the relative deadlines)

In this implementation of the EDF:

At each time iteration T:

1. Scheduler would iterate through the task set, creating a queue of tasks that are currently in ready state sorted by their earliest deadline.
2. Checks if the current task has been completed or if any tasks has missed its deadline.
3. Scheduler checks if the current task ID is different from the task ID at the head of the queue that has just been assigned.
4. Runs chosen task for iteration T.

Since at each time iteration T, the task with the earliest deadline is being chosen, this algorithm automatically accounts for preemption. An additional task with task ID = -1 is required to signify that the scheduler is in idle state. This case only occurs when schedulability is less than one.

EFDF

Like the implementation of EDF, a schedulability test is also calculated before tasks in the task sets are being generated. The general flow of EFDF is like EDF which some changes in the decision making of task priority. In addition to selecting the task with the earliest deadline in the ready queue, it also calculates the laxities of the tasks in the ready queue. The equation to calculate laxity is as follows:

$$L_i(t) = D_i(t) - E_i(t)$$

$D_i(t)$ → deadline by which task ***T_i*** must be completed.

$E_i(t)$ → amount of computation remaining to be performed.

In this implementation of the EFDF:

At each time iteration T:

1. Scheduler would iterate through the task set, creating a queue of tasks that are currently in ready state sorted by their earliest deadline.
2. Checks if the current task has been completed or if any tasks has missed its deadline.
3. Scheduler calculates the laxity of all tasks in the ready queue. If there exists a task in the queue that has zero laxity, it would be elevated to the highest priority and shifted to the head of the queue.
4. Scheduler checks if the current task ID is different from the task ID at the head of the queue that has just been assigned.
5. Runs chosen task for iteration T.

RMS

The RMS algorithm is very similar to the implementation of the EDF with only minor differences. The schedulability test of the RMS algorithm follows this equation:

$$U = \sum_{i=1}^{n} {C_i}/{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

C

$\boldsymbol{C_i} \rightarrow$ worst-case computation times of the n processes.

$\boldsymbol{T_i} \rightarrow$ period of task *i*.

A rough estimate is that the RMS can meet all of the deadlines if CPU utilization is less than 69.32% [1]. Since the RMS algorithm assigns static priorities according to the period of the task, the queue of ready tasks needs to be sorted according to its period instead of its next deadline.

Therefore, in this implementation of RMS:

At each time iteration T:

1. Scheduler would iterate through the task set, creating a queue of tasks that are currently in ready state sorted by their period.
2. Checks if the current task has been completed or if any tasks has missed its deadline.
3. Scheduler checks if the current task ID is different from the task ID at the head of the queue that has just been assigned.
4. Runs chosen task for iteration T.

RMZL

The RMZL implementation is like EFDF's implementation, following the same formula to calculate the laxity of each task in the ready queue, sort by period instead of earliest deadline. However, it is interesting to note that with this additional zero laxity condition, the RMS algorithm, originally being a static-priority scheduling class, has also a dynamic-priority discipline incorporated in the RMZL. [1]

In this implementation of the RMZL:

At each time iteration T:

1. Scheduler would iterate through the task set, creating a queue of tasks that are currently in ready state sorted by their period.
2. Checks if the current task has been completed or if any tasks has missed its deadline.
3. Scheduler calculates the laxity of all tasks in the ready queue. If there exists a task in the queue that has zero laxity, it would be elevated to the highest priority and shifted to the head of the queue.
4. Scheduler checks if the current task ID is different from the task ID at the head of the queue that has just been assigned.
5. Runs chosen task for iteration T.

Simulation & Results

In this simulation, as mentioned in the Input Data section, there will be three different tasks sets that will be tested on all four algorithms. EDF based algorithms and RMS based algorithms would have 2 different small task sets to show its correctness as well as its differences from its original algorithms. The large task set would be shared across EFDF and RMZL for a simple comparison. A more detailed simulation report is included in the appendix at the end of the report to observe what tasks have reached zero laxity and which tasks have been missed.

EDF vs EFDF

Small (U <= 1)

Figure 2 below displays the results when EDF and EFDF are run with the *edf_small_taskset_equal1.txt* task set.

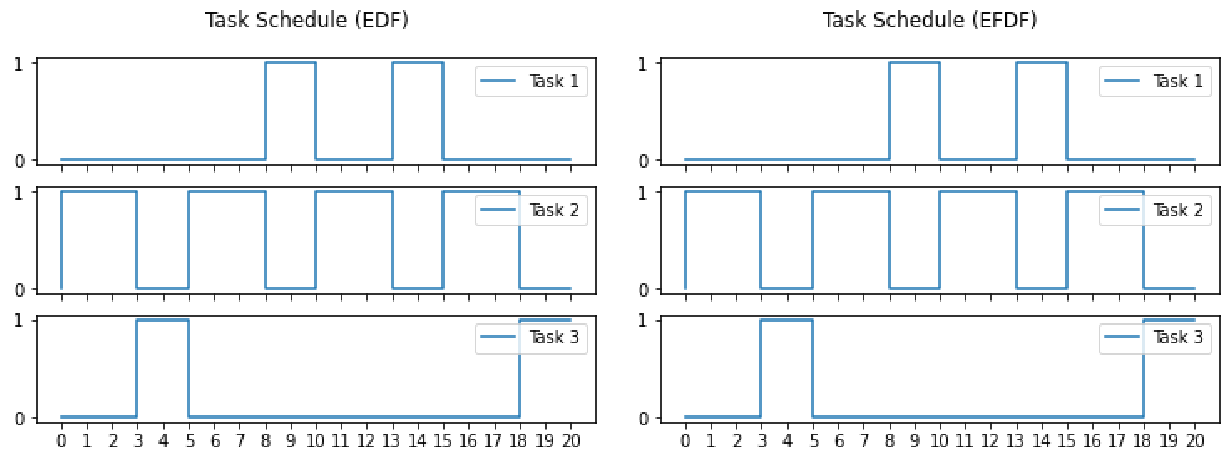


Figure 2. Task Schedule when U <=1 (EDF vs EFDF)

As seen in Figure 2, when the task set has a utilization of less than or equal to one, the result is similar and uninteresting as EDF guarantees all tasks can be scheduled when U<=1.

Small (U > 1)

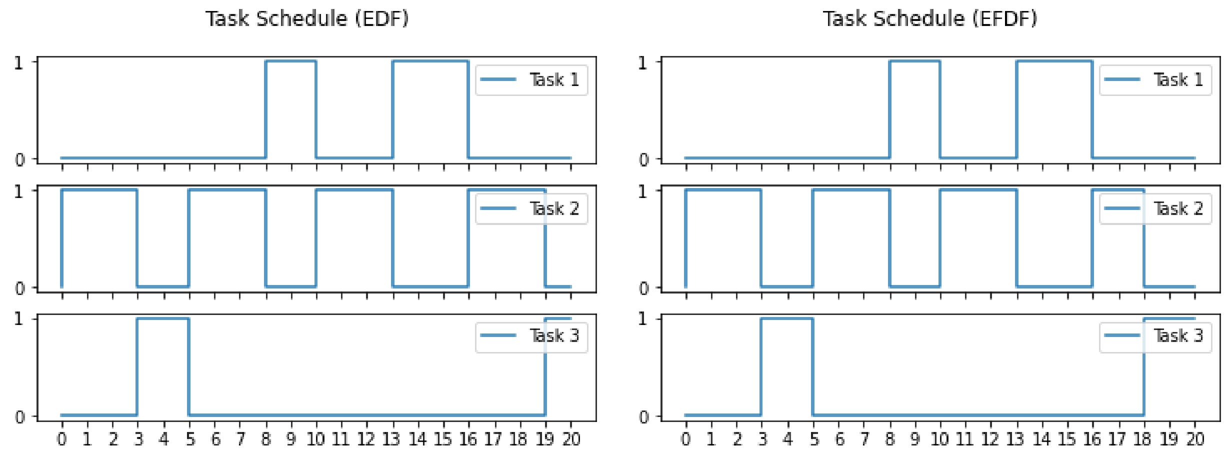


Figure 3. Task Schedule when U>1 (EDF vs EFDF)

Figure 3 above displays the results when *edf_small_taskset_above1.txt* task set is run. The utilization of this task set is 1.05. In Figure 3, there is a deviation of the task schedule between T=18 and T=19. This is because when T=18, Task 3 has fulfilled the zero-laxity condition. According to the assumption 5 in the Input Data section, task 3 now has the highest priority and cannot be preempted until it is completed. In terms of tasks missed, both algorithms are unable to schedule all tasks, and both have 1 missed task each. It is also interesting to note that at T=19, Task 2 would also have reached its zero-laxity condition. However, in this scenario, assumption 5 would take place and task 2 would not preempt task 3.

RMS vs RMZL

Small (U <= 1)

Figure 4 below displays the results when *rms_small_taskset_equal1.txt* task set is run. The utilization bound of this task set is 1.0. Since the RMS has a utilization bound of 0.7798, there may be some tasks that may not be able to meet the deadline.

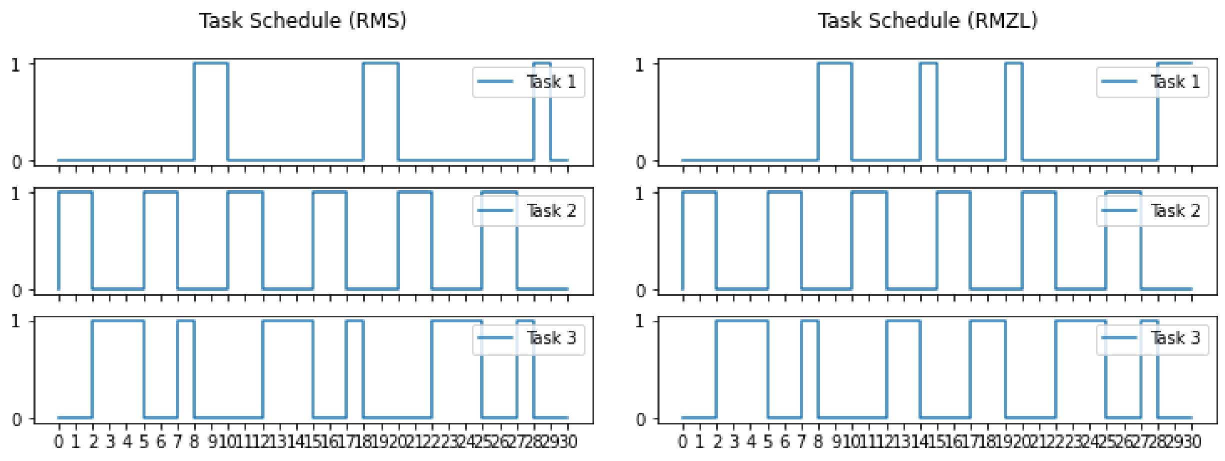


Figure 4. Task Schedule when U<=1 (RMS vs RMZL)

According to the simulation report and Figure 4, the RMS algorithm task 1 has missed its deadline when T=15. This is because at T=10, both Task 2 and Task 3 has been queued up having higher priority since Task 1 has a larger period of 15. As a result, Task 1 has been preempted at T=10 and unable to meet its deadline at T=15. However, in the RMZL algorithm, at T=14, Task 1 has reached zero laxity condition and therefore has the highest priority, resulting in the successfully completed of Task 1. This example supports the claim made in the selected paper that the RMZL algorithm can maintain the system to be as schedulable as much as the EFDF algorithm.

Small (U > 1)

Figure 5 below displays the results when *rms_small_taskset_above_1.txt* task set is run. The utilization bound is 1.100 and since it is outside the RMS utilization bound of 0.7798 and EDF utilization bound of 1, it is expected that both RMS and RMZL algorithms would have some tasks that are unable to meet its deadline.

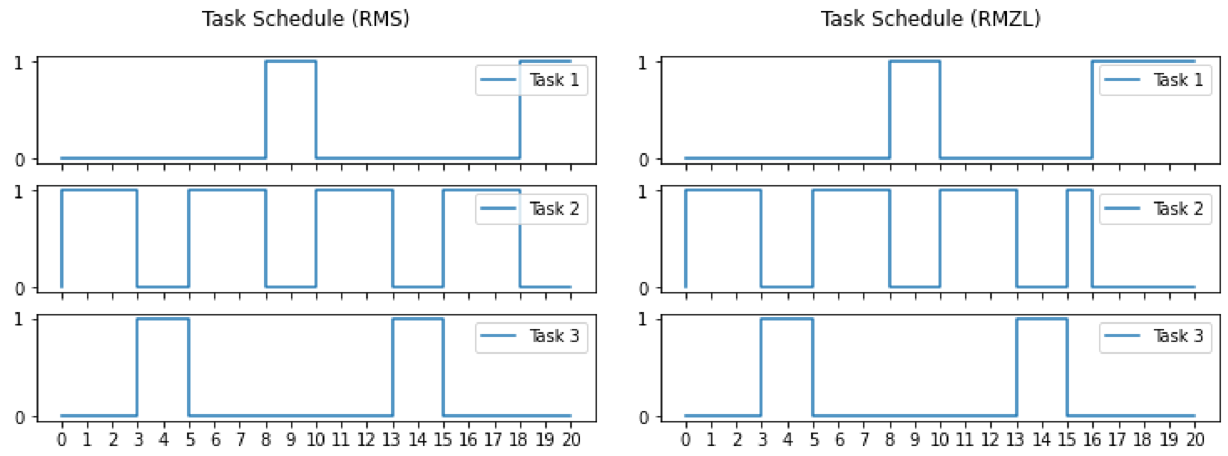


Figure 5. Task Schedule when U>1 (RMS vs RMZL)

As expected, RMS missed Task 1 while RMZL missed Task 2.

EFDF vs RMZL

Large (U > 1)

Figure 6 below depicts the results when *large_taskset.txt* task set is run. This task set contains a total of 7 tasks. The utilization of this task set is 1.68.

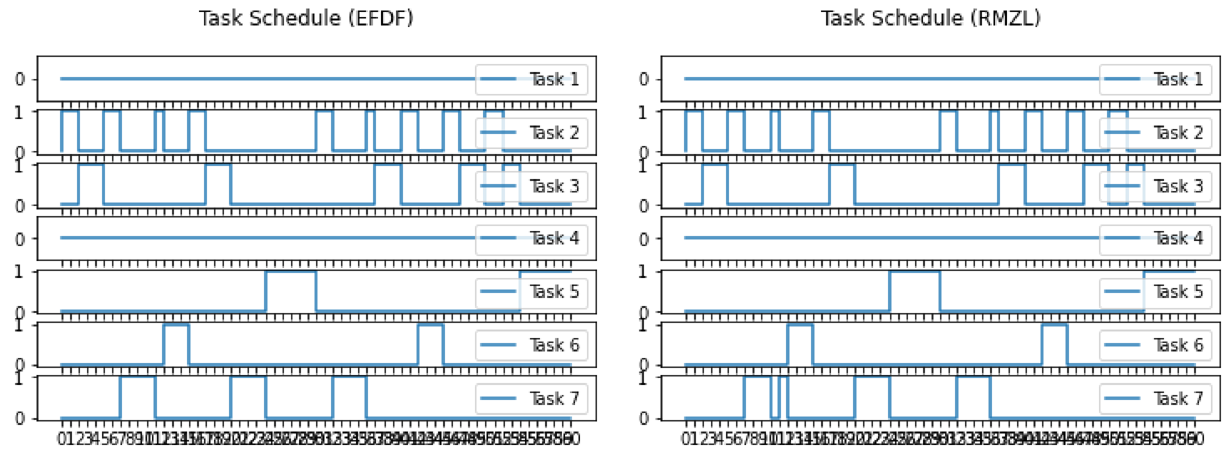


Figure 6. Task Schedule when U>1 (EFDF vs RMZL)

According to the simulation report, both EFDF and RMZL both missed a total of 15 tasks and completed 18 tasks. Both algorithms produce similar results and performance. However, when comparing the number of preemptions, the EFDF has 2 less preemptions as compared to the RMZL algorithm. This could be due to the existence of a very short periodic task in the task set as the RMZL generally schedules based on the shortest period. It is also interesting to note that both Task 1 and 4 did not run at all throughout the hyper period. This is because Task 1 has a very long deadline and period and EFDF and RMZL considers this task to be of extremely low priority. On the contrary, Task 4 has the shortest execution time of only 1 iteration, and it is constantly being starved due to other tasks that are already in the zero-laxity condition.

Strengths and Pitfalls

As shown in the simulation results above, these simulations are built hugely on assumptions that are not realistic in a real-world scenario. The performance of these algorithms also greatly depends on implementations of certain scenarios that may happen in the scheduler.

EFDF is like the EDF in situations where U<=1. However, it may perform better in soft real time systems where U>1, as the negative laxity attribute in the task may be used to determine the priority in the task set to try to fulfil delay tasks instead of continuing to schedule tasks normally based on its earliest deadline.

RMZL on the other hand, seems to perform better than RMS generally. As RMZL now has an additional zero laxity condition, it can dynamically change the priority of a task, causing its utilization bound to increase. However, though it can perform similarly to EDF/EFDF, it may not be as efficient as EDF/EFDF as it schedules the tasks based on it periods and if there exist many high frequency tasks in the task set, it may lead to many preemptions causing a larger amount of overhead.

However, both EFDF and RMZL has its pitfalls as well. Both research papers assumed that when a task has reached zero laxity, the task would be raised to the highest priority and will not be preempted until completed. This could be a potential flaw in the algorithm as it could lead to resource starvation for other tasks as they are no longer able to preempt the zero-laxity task.

Therefore, it is imperative that some measure should be in place to handle these conditions. One possible solution in a soft real time system could be checking the task’s own priority if there are more than 2 tasks in the zero-laxity condition. The task with the higher priority task would then be scheduled.

Conclusion

In conclusion, it is imperative to figure out the requirements and needs for a real time system before deciding on the algorithm. Though the RMZL and EFDF algorithms seems to be more versatile and efficient compared to its original algorithm, it has also brought more complications when dealing with tasks in zero-laxity conditions. This may not be desired in certain real time systems where predictability and simplicity are more important. In addition, when using EFDF and RMZL, it is extremely important to define individual tasks priorities to deal with situations where there are more than 1 task in the zero-laxity condition.

References

[1] Kato, S. et al. “Global Rate-Monotonic Scheduling with Priority Promotion.” (2008).

[2] agbeer Singh, Bichitrananda Patra and Satyendra Prasad Singh, “An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System ” International Journal of Advanced Computer Science and Applications(IJACSA), 2(2), 2011. http://dx.doi.org/10.14569/IJACSA.2011.020207

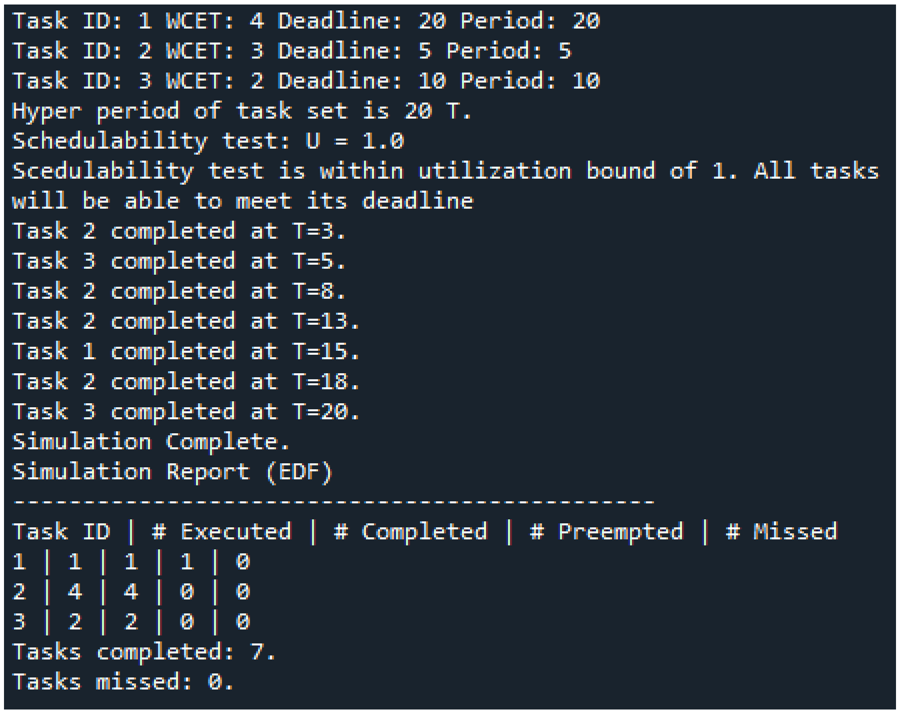
Coding Effort

All code is written 100% by me. I did not refer to any source codes from github or any other sites.

Data generated from my experiments are generated by me. No external tools or third party software are used to generate the task sets.

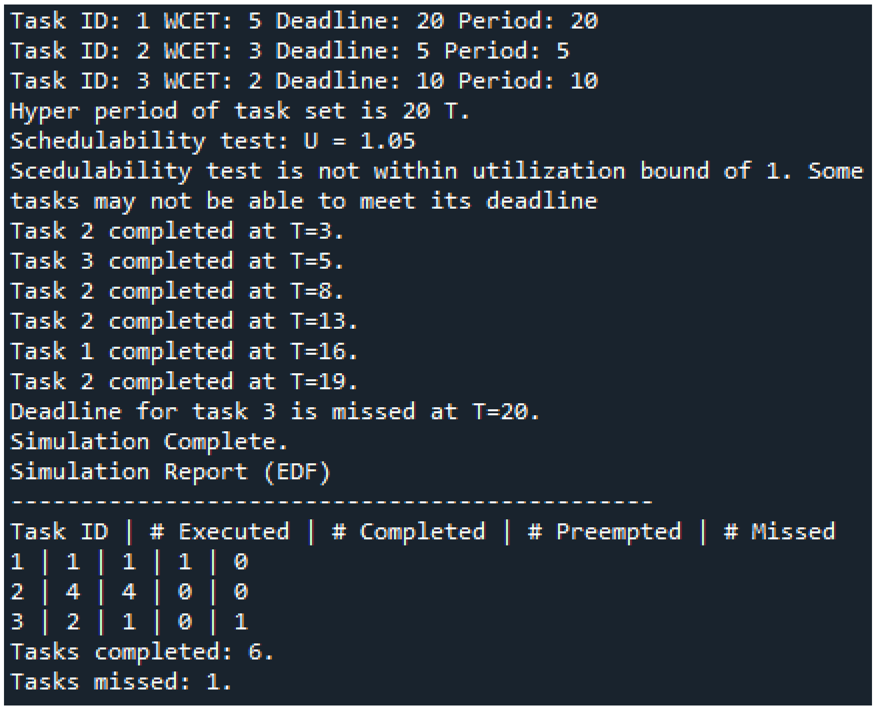
Appendix

EDF Simulation Report (edf_small_taskset_equal1.txt)



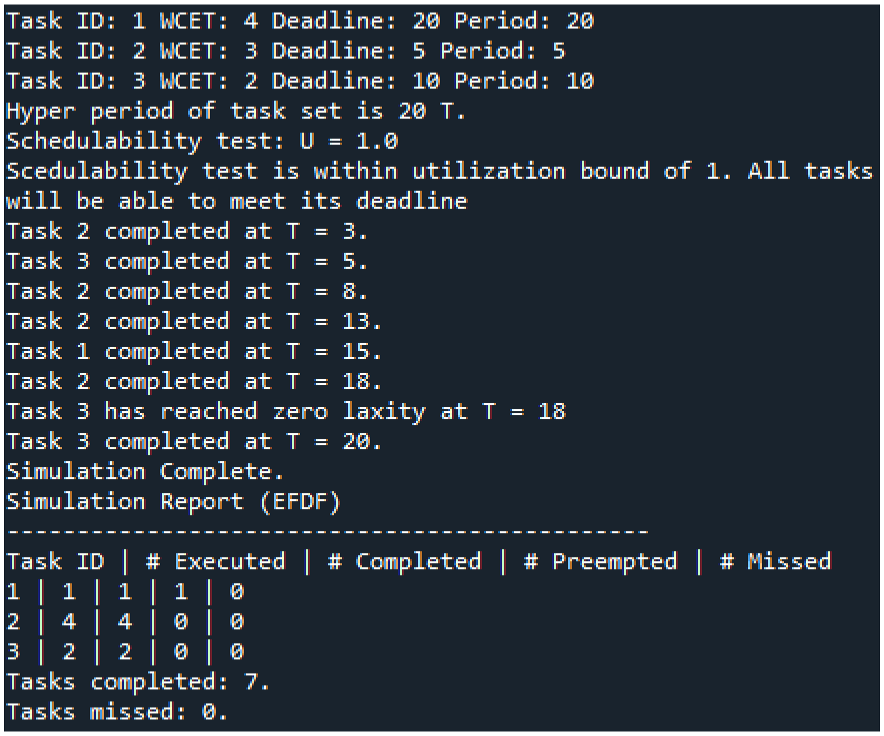
```
Task ID: 1 WCET: 4 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.0
Scedulability test is within utilization bound of 1. All tasks
will be able to meet its deadline
Task 2 completed at T=3.
Task 3 completed at T=5.
Task 2 completed at T=8.
Task 2 completed at T=13.
Task 1 completed at T=15.
Task 2 completed at T=18.
Task 3 completed at T=20.
Simulation Complete.
Simulation Report (EDF)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 1 | 1 | 0
2 | 4 | 4 | 0 | 0
3 | 2 | 2 | 0 | 0
Tasks completed: 7.
Tasks missed: 0.
```

EDF Simulation Report (edf_small_taskset_above1.txt)



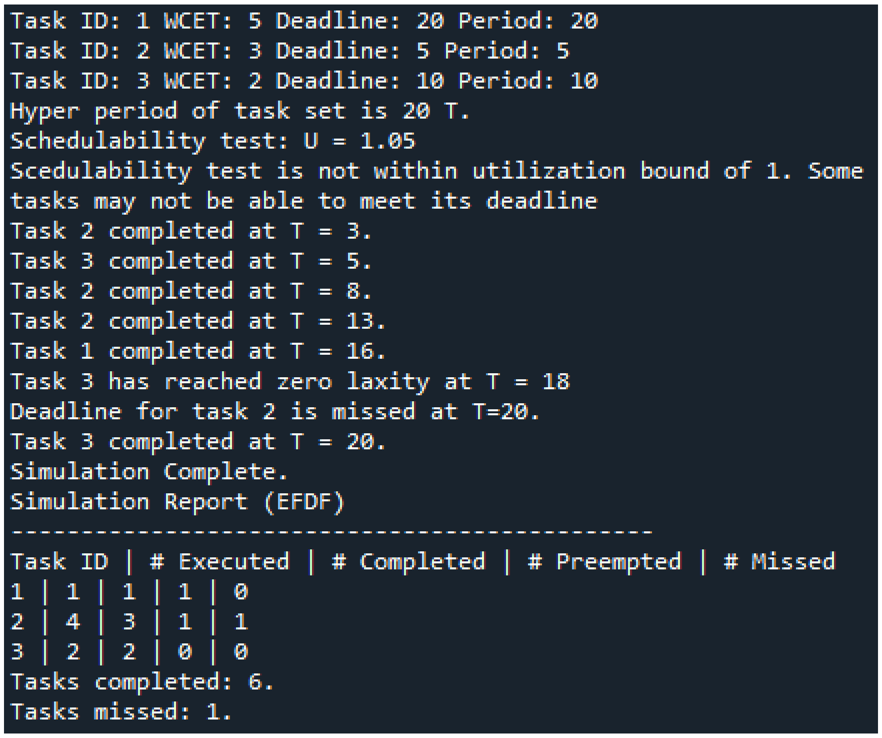
```
Task ID: 1 WCET: 5 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.05
Scedulability test is not within utilization bound of 1. Some
tasks may not be able to meet its deadline
Task 2 completed at T=3.
Task 3 completed at T=5.
Task 2 completed at T=8.
Task 2 completed at T=13.
Task 1 completed at T=16.
Task 2 completed at T=19.
Deadline for task 3 is missed at T=20.
Simulation Complete.
Simulation Report (EDF)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 1 | 1 | 0
2 | 4 | 4 | 0 | 0
3 | 2 | 1 | 0 | 1
Tasks completed: 6.
Tasks missed: 1.
```

EFDF Simulation Report (edf_small_taskset_equal1.txt)



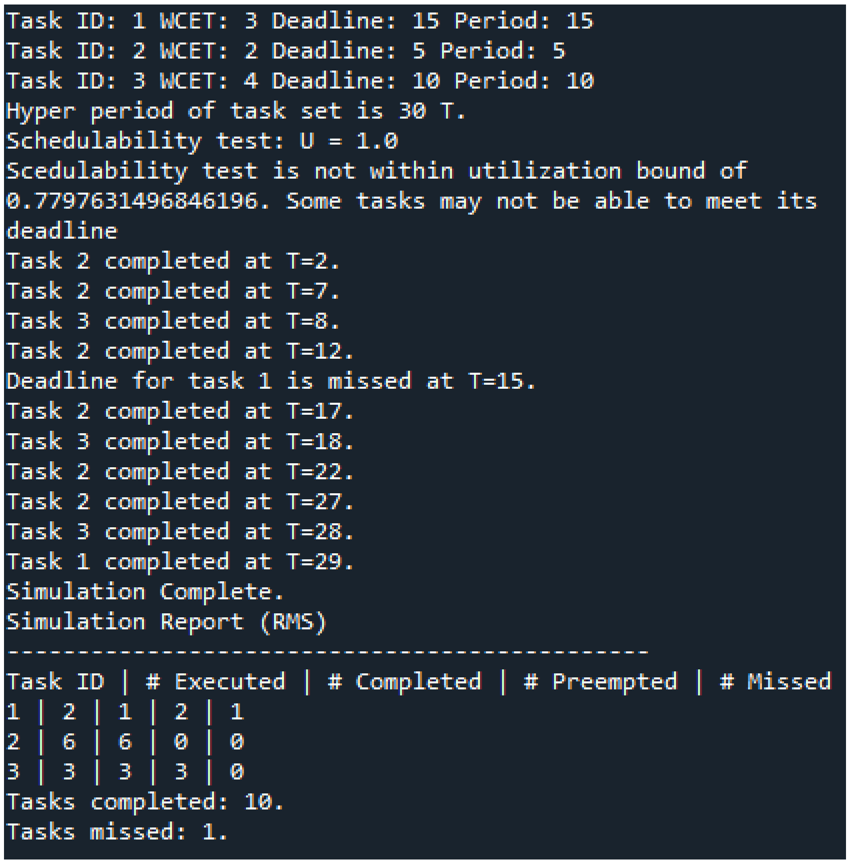
```
Task ID: 1 WCET: 4 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.0
Scedulability test is within utilization bound of 1. All tasks
will be able to meet its deadline
Task 2 completed at T = 3.
Task 3 completed at T = 5.
Task 2 completed at T = 8.
Task 2 completed at T = 13.
Task 1 completed at T = 15.
Task 2 completed at T = 18.
Task 3 has reached zero laxity at T = 18
Task 3 completed at T = 20.
Simulation Complete.
Simulation Report (EFDF)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 1 | 1 | 0
2 | 4 | 4 | 0 | 0
3 | 2 | 2 | 0 | 0
Tasks completed: 7.
Tasks missed: 0.
```

EFDF Simulation Report (edf_small_taskset_above1.txt)



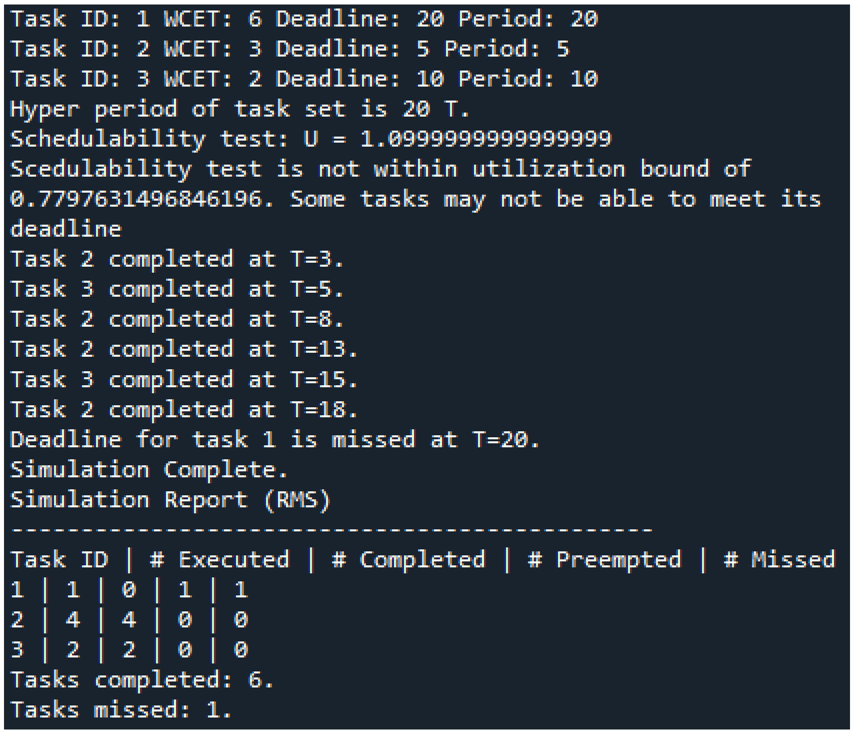
```
Task ID: 1 WCET: 5 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.05
Scedulability test is not within utilization bound of 1. Some
tasks may not be able to meet its deadline
Task 2 completed at T = 3.
Task 3 completed at T = 5.
Task 2 completed at T = 8.
Task 2 completed at T = 13.
Task 1 completed at T = 16.
Task 3 has reached zero laxity at T = 18
Deadline for task 2 is missed at T=20.
Task 3 completed at T = 20.
Simulation Complete.
Simulation Report (EFDF)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 1 | 1 | 0
2 | 4 | 3 | 1 | 1
3 | 2 | 2 | 0 | 0
Tasks completed: 6.
Tasks missed: 1.
```

RMS Simulation Report (rms_small_taskset_equal1.txt)



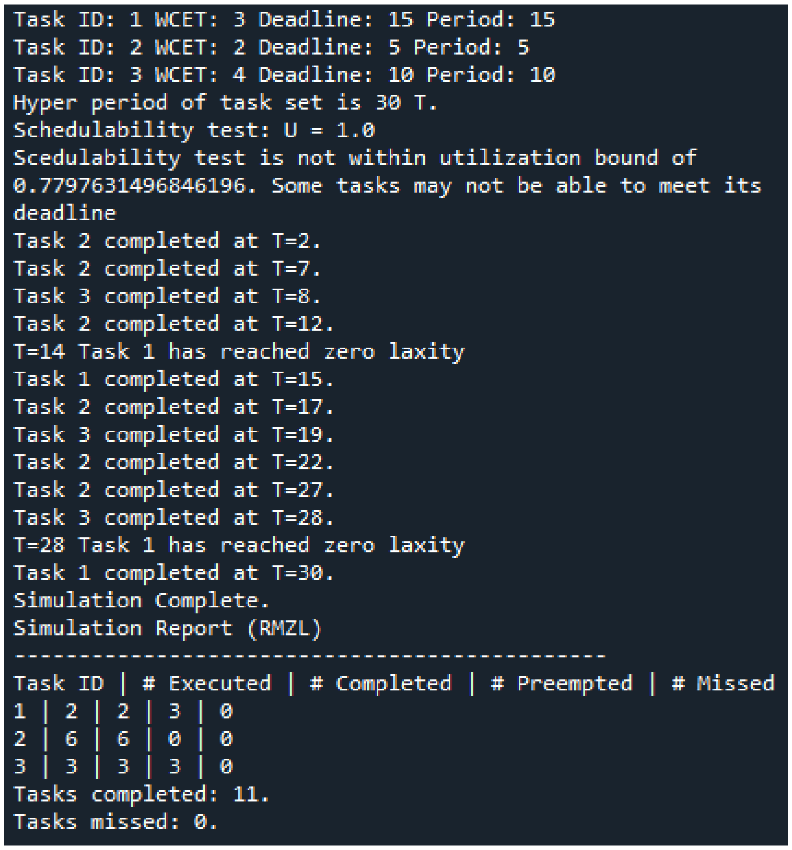
```
Task ID: 1 WCET: 3 Deadline: 15 Period: 15
Task ID: 2 WCET: 2 Deadline: 5 Period: 5
Task ID: 3 WCET: 4 Deadline: 10 Period: 10
Hyper period of task set is 30 T.
Schedulability test: U = 1.0
Scedulability test is not within utilization bound of
0.7797631496846196. Some tasks may not be able to meet its
deadline
Task 2 completed at T=2.
Task 2 completed at T=7.
Task 3 completed at T=8.
Task 2 completed at T=12.
Deadline for task 1 is missed at T=15.
Task 2 completed at T=17.
Task 3 completed at T=18.
Task 2 completed at T=22.
Task 2 completed at T=27.
Task 3 completed at T=28.
Task 1 completed at T=29.
Simulation Complete.
Simulation Report (RMS)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 2 | 1 | 2 | 1
2 | 6 | 6 | 0 | 0
3 | 3 | 3 | 3 | 0
Tasks completed: 10.
Tasks missed: 1.
```

RMS Simulation Report (rms_small_taskset_above1.txt)



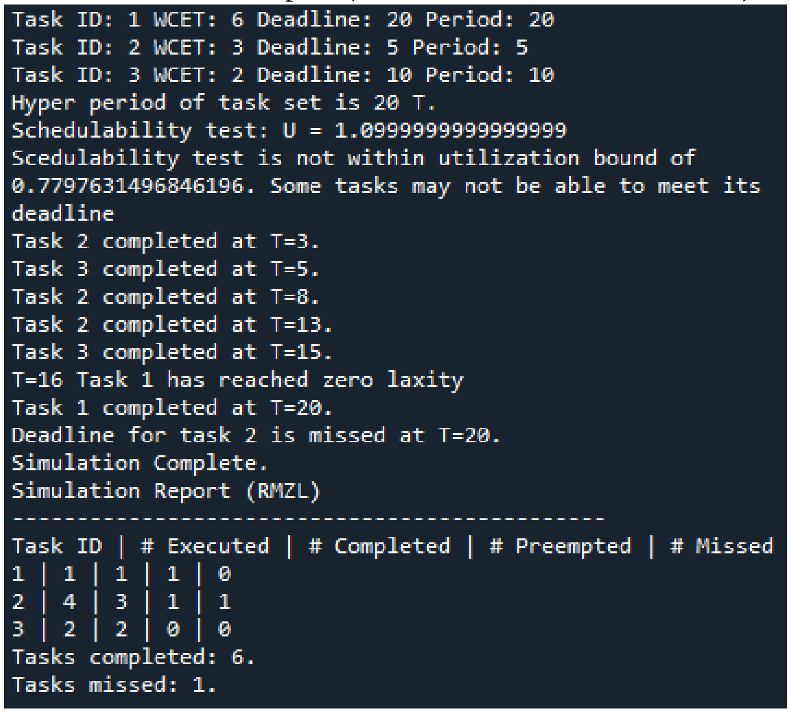
```
Task ID: 1 WCET: 6 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.0999999999999999
Scedulability test is not within utilization bound of
0.7797631496846196. Some tasks may not be able to meet its
deadline
Task 2 completed at T=3.
Task 3 completed at T=5.
Task 2 completed at T=8.
Task 2 completed at T=13.
Task 3 completed at T=15.
Task 2 completed at T=18.
Deadline for task 1 is missed at T=20.
Simulation Complete.
Simulation Report (RMS)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 0 | 1 | 1
2 | 4 | 4 | 0 | 0
3 | 2 | 2 | 0 | 0
Tasks completed: 6.
Tasks missed: 1.
```

RMZL Simulation Report (rms_small_taskset_equal1.txt)



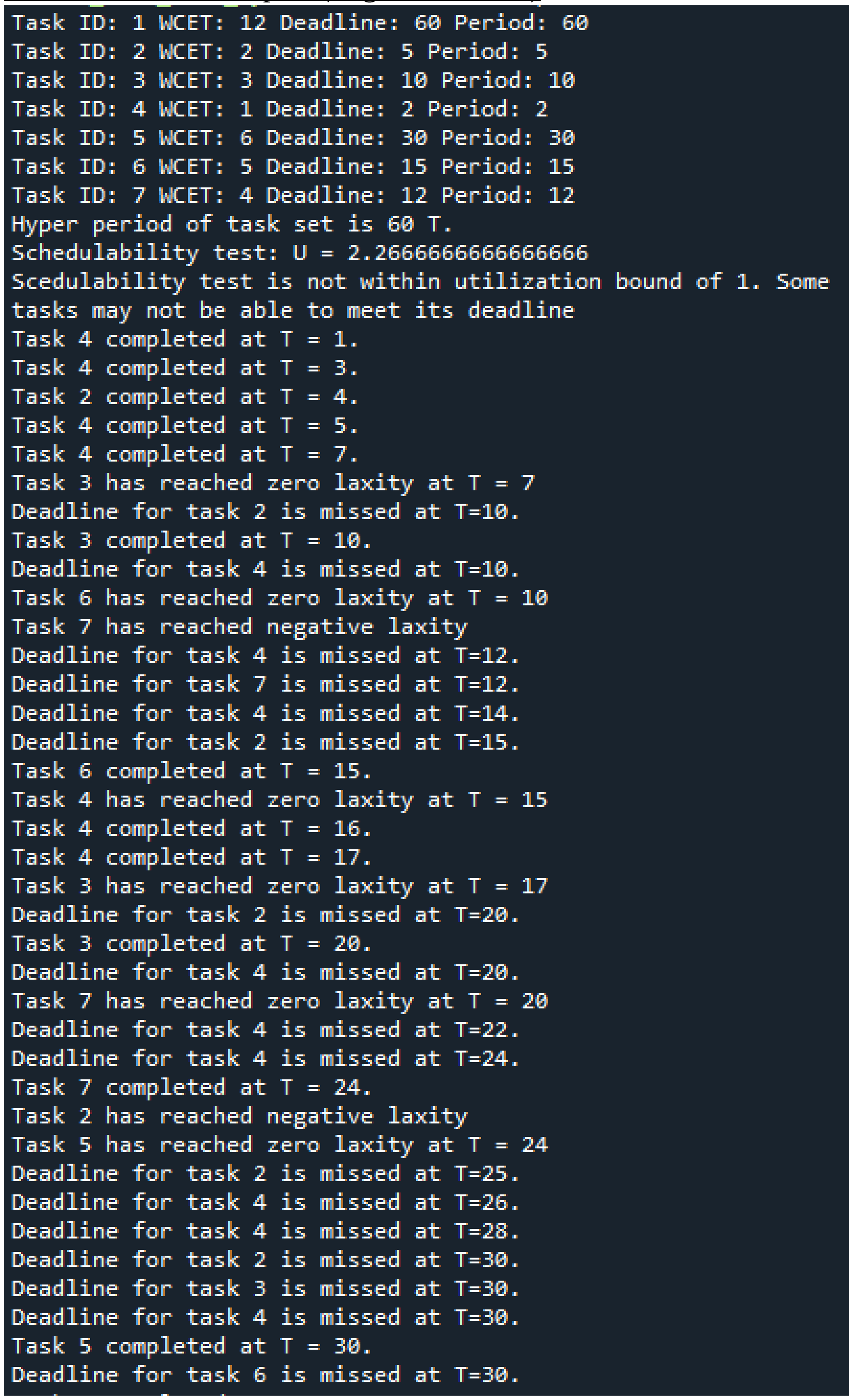
```
Task ID: 1 WCET: 3 Deadline: 15 Period: 15
Task ID: 2 WCET: 2 Deadline: 5 Period: 5
Task ID: 3 WCET: 4 Deadline: 10 Period: 10
Hyper period of task set is 30 T.
Schedulability test: U = 1.0
Scedulability test is not within utilization bound of
0.7797631496846196. Some tasks may not be able to meet its
deadline
Task 2 completed at T=2.
Task 2 completed at T=7.
Task 3 completed at T=8.
Task 2 completed at T=12.
T=14 Task 1 has reached zero laxity
Task 1 completed at T=15.
Task 2 completed at T=17.
Task 3 completed at T=19.
Task 2 completed at T=22.
Task 2 completed at T=27.
Task 3 completed at T=28.
T=28 Task 1 has reached zero laxity
Task 1 completed at T=30.
Simulation Complete.
Simulation Report (RMZL)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 2 | 2 | 3 | 0
2 | 6 | 6 | 0 | 0
3 | 3 | 3 | 3 | 0
Tasks completed: 11.
Tasks missed: 0.
```

RMZL Simulation Report (rms_small_taskset_above1.txt)

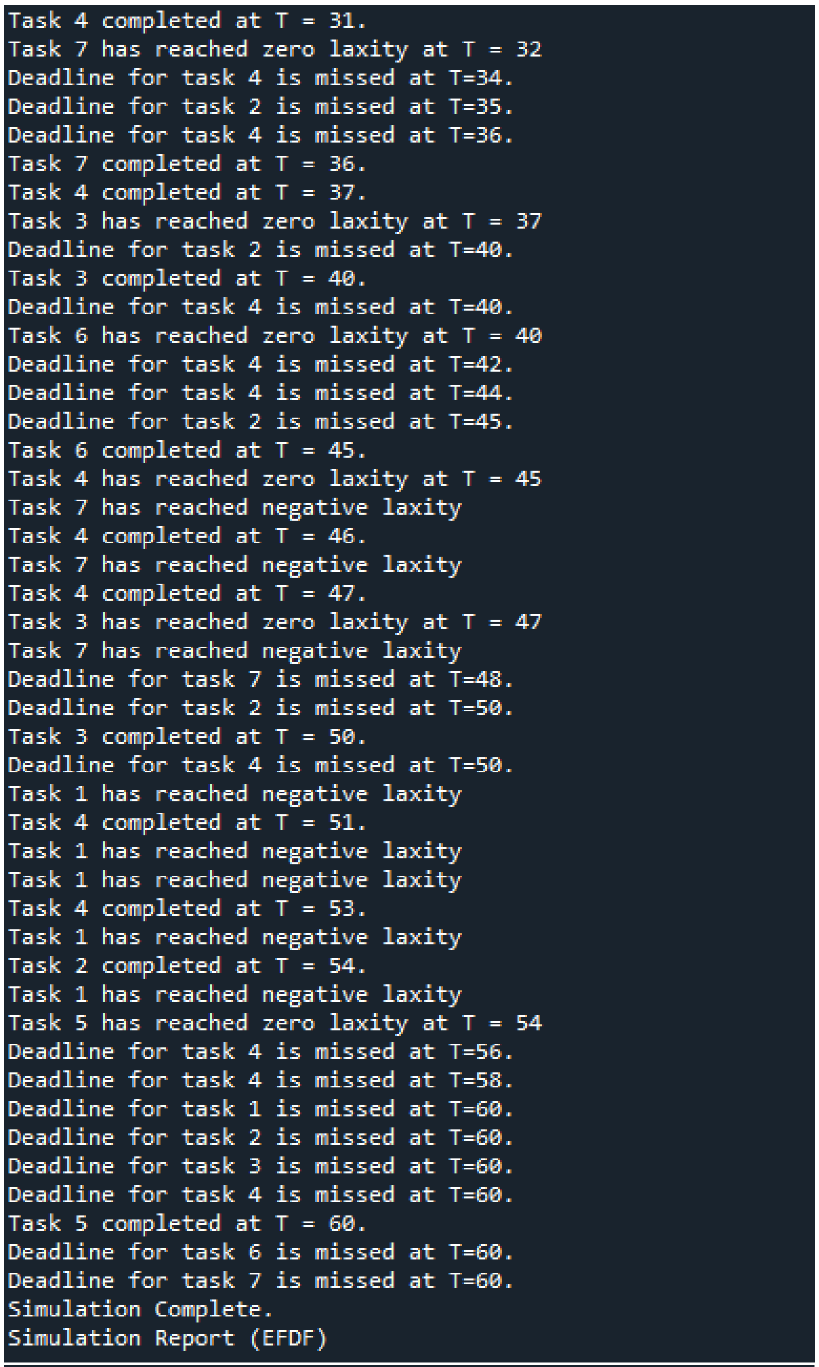


```
Task ID: 1 WCET: 6 Deadline: 20 Period: 20
Task ID: 2 WCET: 3 Deadline: 5 Period: 5
Task ID: 3 WCET: 2 Deadline: 10 Period: 10
Hyper period of task set is 20 T.
Schedulability test: U = 1.0999999999999999
Scedulability test is not within utilization bound of
0.7797631496846196. Some tasks may not be able to meet its
deadline
Task 2 completed at T=3.
Task 3 completed at T=5.
Task 2 completed at T=8.
Task 2 completed at T=13.
Task 3 completed at T=15.
T=16 Task 1 has reached zero laxity
Task 1 completed at T=20.
Deadline for task 2 is missed at T=20.
Simulation Complete.
Simulation Report (RMZL)
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 1 | 1 | 0
2 | 4 | 3 | 1 | 1
3 | 2 | 2 | 0 | 0
Tasks completed: 6.
Tasks missed: 1.
```

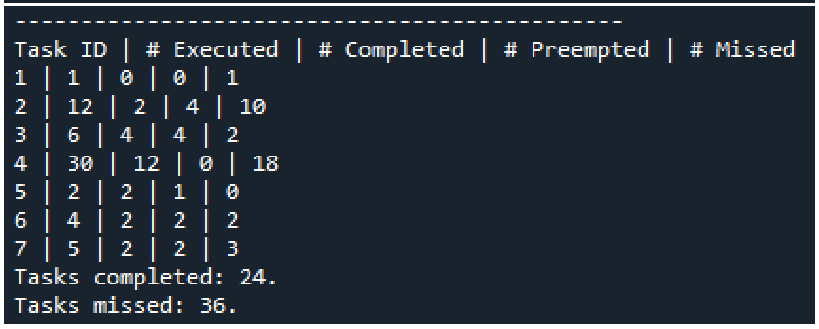
EFDF Simulation Report (large_taskset.txt)



```
Task ID: 1 WCET: 12 Deadline: 60 Period: 60
Task ID: 2 WCET: 2 Deadline: 5 Period: 5
Task ID: 3 WCET: 3 Deadline: 10 Period: 10
Task ID: 4 WCET: 1 Deadline: 2 Period: 2
Task ID: 5 WCET: 6 Deadline: 30 Period: 30
Task ID: 6 WCET: 5 Deadline: 15 Period: 15
Task ID: 7 WCET: 4 Deadline: 12 Period: 12
Hyper period of task set is 60 T.
Schedulability test: U = 2.2666666666666666
Scedulability test is not within utilization bound of 1. Some
tasks may not be able to meet its deadline
Task 4 completed at T = 1.
Task 4 completed at T = 3.
Task 2 completed at T = 4.
Task 4 completed at T = 5.
Task 4 completed at T = 7.
Task 3 has reached zero laxity at T = 7
Deadline for task 2 is missed at T=10.
Task 3 completed at T = 10.
Deadline for task 4 is missed at T=10.
Task 6 has reached zero laxity at T = 10
Task 7 has reached negative laxity
Deadline for task 4 is missed at T=12.
Deadline for task 7 is missed at T=12.
Deadline for task 4 is missed at T=14.
Deadline for task 2 is missed at T=15.
Task 6 completed at T = 15.
Task 4 has reached zero laxity at T = 15
Task 4 completed at T = 16.
Task 4 completed at T = 17.
Task 3 has reached zero laxity at T = 17
Deadline for task 2 is missed at T=20.
Task 3 completed at T = 20.
Deadline for task 4 is missed at T=20.
Task 7 has reached zero laxity at T = 20
Deadline for task 4 is missed at T=22.
Deadline for task 4 is missed at T=24.
Task 7 completed at T = 24.
Task 2 has reached negative laxity
Task 5 has reached zero laxity at T = 24
Deadline for task 2 is missed at T=25.
Deadline for task 4 is missed at T=26.
Deadline for task 4 is missed at T=28.
Deadline for task 2 is missed at T=30.
Deadline for task 3 is missed at T=30.
Deadline for task 4 is missed at T=30.
Task 5 completed at T = 30.
Deadline for task 6 is missed at T=30.
```

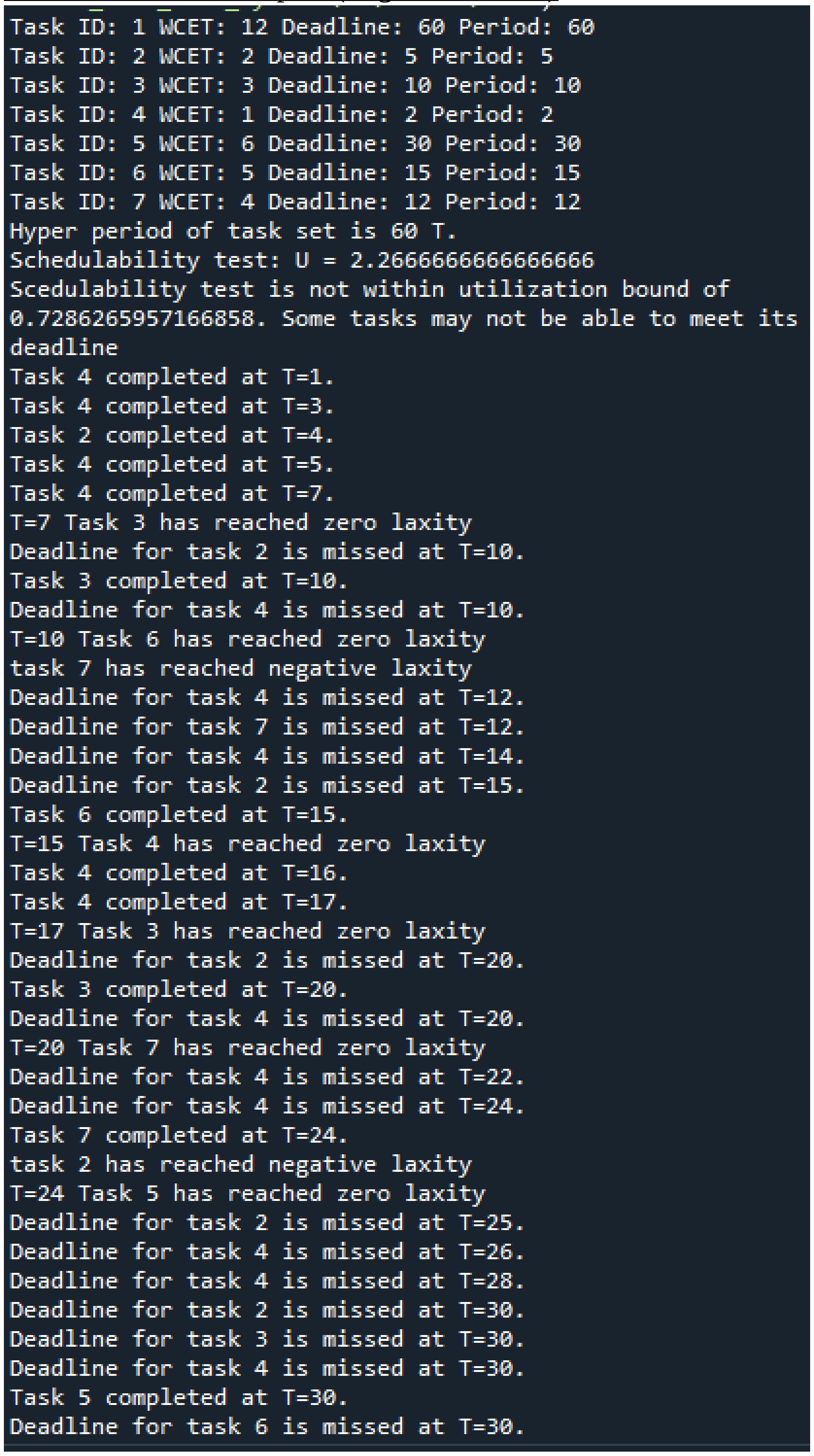


```
Task 4 completed at T = 31.
Task 7 has reached zero laxity at T = 32
Deadline for task 4 is missed at T=34.
Deadline for task 2 is missed at T=35.
Deadline for task 4 is missed at T=36.
Task 7 completed at T = 36.
Task 4 completed at T = 37.
Task 3 has reached zero laxity at T = 37
Deadline for task 2 is missed at T=40.
Task 3 completed at T = 40.
Deadline for task 4 is missed at T=40.
Task 6 has reached zero laxity at T = 40
Deadline for task 4 is missed at T=42.
Deadline for task 4 is missed at T=44.
Deadline for task 2 is missed at T=45.
Task 6 completed at T = 45.
Task 4 has reached zero laxity at T = 45
Task 7 has reached negative laxity
Task 4 completed at T = 46.
Task 7 has reached negative laxity
Task 4 completed at T = 47.
Task 3 has reached zero laxity at T = 47
Task 7 has reached negative laxity
Deadline for task 7 is missed at T=48.
Deadline for task 2 is missed at T=50.
Task 3 completed at T = 50.
Deadline for task 4 is missed at T=50.
Task 1 has reached negative laxity
Task 4 completed at T = 51.
Task 1 has reached negative laxity
Task 1 has reached negative laxity
Task 4 completed at T = 53.
Task 1 has reached negative laxity
Task 2 completed at T = 54.
Task 1 has reached negative laxity
Task 5 has reached zero laxity at T = 54
Deadline for task 4 is missed at T=56.
Deadline for task 4 is missed at T=58.
Deadline for task 1 is missed at T=60.
Deadline for task 2 is missed at T=60.
Deadline for task 3 is missed at T=60.
Deadline for task 4 is missed at T=60.
Task 5 completed at T = 60.
Deadline for task 6 is missed at T=60.
Deadline for task 7 is missed at T=60.
Simulation Complete.
Simulation Report (EFDF)
```

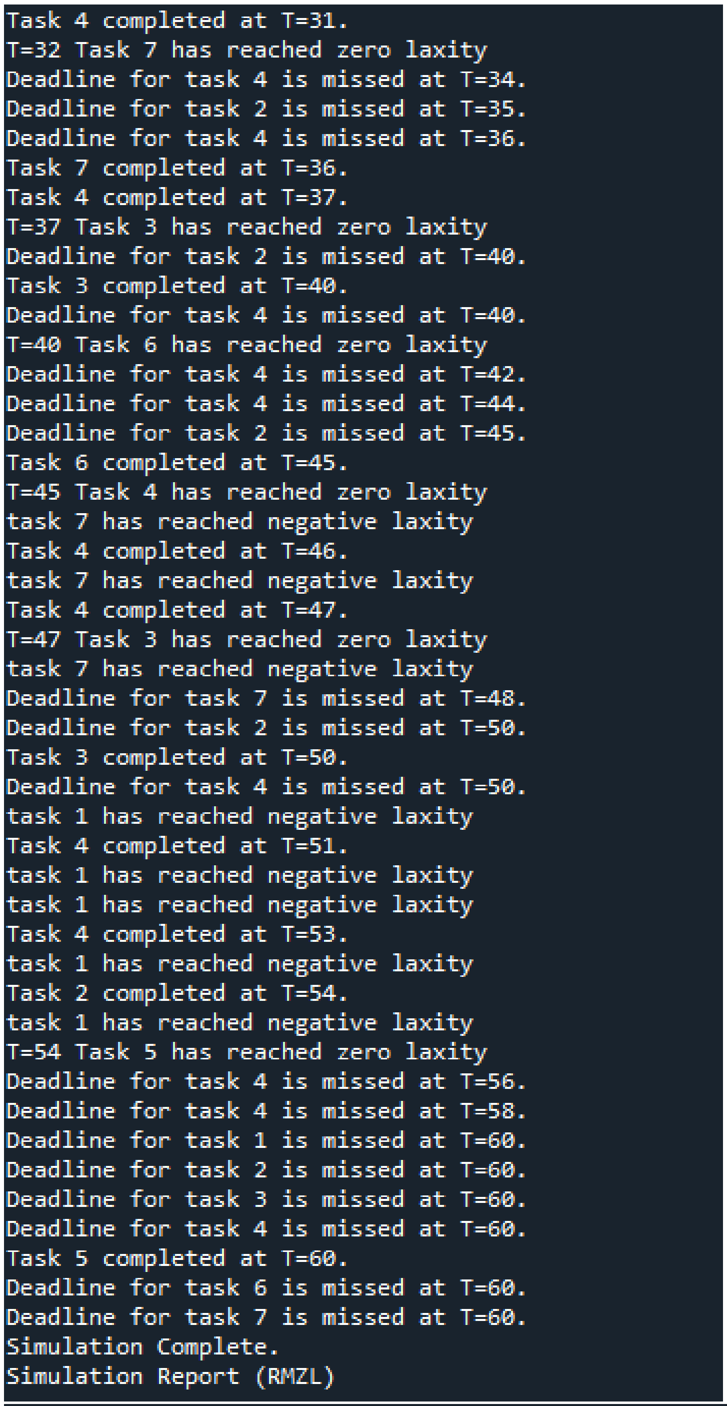


```
----------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 0 | 0 | 1
2 | 12 | 2 | 4 | 10
3 | 6 | 4 | 4 | 2
4 | 30 | 12 | 0 | 18
5 | 2 | 2 | 1 | 0
6 | 4 | 2 | 2 | 2
7 | 5 | 2 | 2 | 3
Tasks completed: 24.
Tasks missed: 36.
```

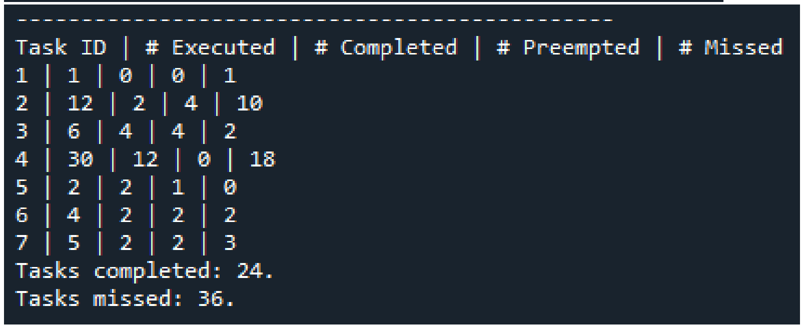
RMZL Simulation Report (large_taskset.txt)



```
Task ID: 1 WCET: 12 Deadline: 60 Period: 60
Task ID: 2 WCET: 2 Deadline: 5 Period: 5
Task ID: 3 WCET: 3 Deadline: 10 Period: 10
Task ID: 4 WCET: 1 Deadline: 2 Period: 2
Task ID: 5 WCET: 6 Deadline: 30 Period: 30
Task ID: 6 WCET: 5 Deadline: 15 Period: 15
Task ID: 7 WCET: 4 Deadline: 12 Period: 12
Hyper period of task set is 60 T.
Schedulability test: U = 2.2666666666666666
Scedulability test is not within utilization bound of
0.7286265957166858. Some tasks may not be able to meet its
deadline
Task 4 completed at T=1.
Task 4 completed at T=3.
Task 2 completed at T=4.
Task 4 completed at T=5.
Task 4 completed at T=7.
T=7 Task 3 has reached zero laxity
Deadline for task 2 is missed at T=10.
Task 3 completed at T=10.
Deadline for task 4 is missed at T=10.
T=10 Task 6 has reached zero laxity
task 7 has reached negative laxity
Deadline for task 4 is missed at T=12.
Deadline for task 7 is missed at T=12.
Deadline for task 4 is missed at T=14.
Deadline for task 2 is missed at T=15.
Task 6 completed at T=15.
T=15 Task 4 has reached zero laxity
Task 4 completed at T=16.
Task 4 completed at T=17.
T=17 Task 3 has reached zero laxity
Deadline for task 2 is missed at T=20.
Task 3 completed at T=20.
Deadline for task 4 is missed at T=20.
T=20 Task 7 has reached zero laxity
Deadline for task 4 is missed at T=22.
Deadline for task 4 is missed at T=24.
Task 7 completed at T=24.
task 2 has reached negative laxity
T=24 Task 5 has reached zero laxity
Deadline for task 2 is missed at T=25.
Deadline for task 4 is missed at T=26.
Deadline for task 4 is missed at T=28.
Deadline for task 2 is missed at T=30.
Deadline for task 3 is missed at T=30.
Deadline for task 4 is missed at T=30.
Task 5 completed at T=30.
Deadline for task 6 is missed at T=30.
```



```
Task 4 completed at T=31.
T=32 Task 7 has reached zero laxity
Deadline for task 4 is missed at T=34.
Deadline for task 2 is missed at T=35.
Deadline for task 4 is missed at T=36.
Task 7 completed at T=36.
Task 4 completed at T=37.
T=37 Task 3 has reached zero laxity
Deadline for task 2 is missed at T=40.
Task 3 completed at T=40.
Deadline for task 4 is missed at T=40.
T=40 Task 6 has reached zero laxity
Deadline for task 4 is missed at T=42.
Deadline for task 4 is missed at T=44.
Deadline for task 2 is missed at T=45.
Task 6 completed at T=45.
T=45 Task 4 has reached zero laxity
task 7 has reached negative laxity
Task 4 completed at T=46.
task 7 has reached negative laxity
Task 4 completed at T=47.
T=47 Task 3 has reached zero laxity
task 7 has reached negative laxity
Deadline for task 7 is missed at T=48.
Deadline for task 2 is missed at T=50.
Task 3 completed at T=50.
Deadline for task 4 is missed at T=50.
task 1 has reached negative laxity
Task 4 completed at T=51.
task 1 has reached negative laxity
task 1 has reached negative laxity
Task 4 completed at T=53.
task 1 has reached negative laxity
Task 2 completed at T=54.
task 1 has reached negative laxity
T=54 Task 5 has reached zero laxity
Deadline for task 4 is missed at T=56.
Deadline for task 4 is missed at T=58.
Deadline for task 1 is missed at T=60.
Deadline for task 2 is missed at T=60.
Deadline for task 3 is missed at T=60.
Deadline for task 4 is missed at T=60.
Task 5 completed at T=60.
Deadline for task 6 is missed at T=60.
Deadline for task 7 is missed at T=60.
Simulation Complete.
Simulation Report (RMZL)
```



```
---------------------------------------------
Task ID | # Executed | # Completed | # Preempted | # Missed
1 | 1 | 0 | 0 | 1
2 | 12 | 2 | 4 | 10
3 | 6 | 4 | 4 | 2
4 | 30 | 12 | 0 | 18
5 | 2 | 2 | 1 | 0
6 | 4 | 2 | 2 | 2
7 | 5 | 2 | 2 | 3
Tasks completed: 24.
Tasks missed: 36.
```