Department of Computer Science BOSTON UNIVERSITY

CS-350 - Fundamentals of Computing Systems

Midterm Exam #2
Fall 2022

Name: Jae Hong Lee

BU Username: jhonglee BU ID: U27565203

NOTE: *Please use only the provided space and the included extra pages to answer the questions. If you do use the extra pages, make sure you reference them properly in your solutions.*

Problem #1:	24	/24
Problem #2:	6	/22
Problem #3:	14	/22
Problem #4:	14	/22
Problem #5:	12	/20
Total Score:	70	/110

Remarks:

- This is a closed-book/notes exam.
- Basic calculators are allowed.
- You have 80 minutes to complete your exam.
- There are 110 total points.
- If your score is more than 100, it is capped at 100.
- Show your work for full marks.
- Problems and sub-problems weighted as shown.
- **Explain all your assumptions clearly.**

Problem 1

Label each of the statements below with either **True (T)** or **False (F)**:

	Statement	T/F
a.	In Round-Robin scheduling, if the quantum size is too small, too many preemptions might be caused, leading to high overhead.	T
b.	Two systems A and B have identical arrival and service characteristics. A uses SJN to schedule requests; B uses FIFO. The average response time at A is higher than B.	F
c.	When scheduling a stateful resource, a work-conserving scheduling strategy might significantly impact the utilization of the resource.	T
d.	The Completely Fair Queuing (CFQ) disk scheduling strategy does not perform request reordering to reduce disk head movements.	F
e.	The Shortest Remaining Time (SRT) scheduling strategy can be considered a preemptive variant of SJN	T
f.	Unlike Rate Monotonic (RM), Earliest Deadline First (EDF) is a non-preemptive scheduling strategy.	F
g.	It is possible that a real-time taskset with 100% utilization is schedulable using SJN on a single-processor CPU.	T
h.	Highest Slowdown Next (HSN) ensures that no pending job suffers starvation as long as no job of infinite length exists in the system.	T
i.	The worst-case response time (WCRT) of job in a real-time taskset is not impacted by the scheduling strategy, but only by the parameters of the tasks.	F
j.	Despite the higher utilization bound of EDF compared to RM, RM is more desirable for real-world safety-critical systems thanks to its simplicity.	T
k.	If a valid task-to-CPU assignment is produced by EDF-FF on a given taskset, then there exists a partitioned EDF strategy capable of scheduling the taskset.	T
l.	Multi-threaded code that uses semaphores for synchronization might introduce deadlocks if the semaphores are not properly initialized and used in the code.	T
m.	The Peterson's Algorithm for 2-party mutual exclusion requires hardware-level support for atomic `test_and_set` instructions to work on in-order CPUs.	T
n.	First-Ready, First-Come First-Served (FR-FCFS) ensures that no starvation is suffered by pending memory requests.	F
o.	Global EDF is an optimal scheduling strategy for real-time tasks on a 2-CPU system.	F
p.	If RM-FF is able to produce a valid task-to-processor assignment, then EDF-FF will also produce a valid assignment when considering tasks in the same order.	T
q.	There is no need to worry about mutual exclusion on a single-processor in-order CPU.	T
r.	A spinlock allows mutual exclusion even with $N > 2$ processes on a multi-processor out-of-order CPU	T

Note: There are 18 questions. A correct answer will get you 2 points; an incorrect answer -1 points; a blank answer 0 points. The final score is capped at 24.

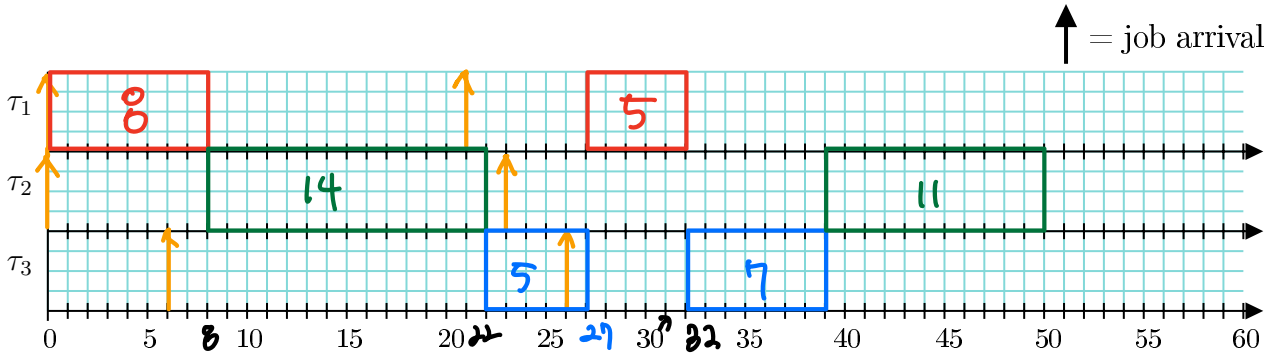
Problem 2

Three tasks τ_1, τ_2, and τ_3 are being scheduled on a single-processor CPU using the Highest Slowdown Next (HSN) scheduling strategy. Any tie is broken by scheduling the ready job that belongs to the task with the lowest ID. Consider that the schedule starts at time 0. At time 0, you have already observed the length of three jobs of each task that have already been completed. Recall the notation $C_{i,j}$ refers to the length (C) of the j-th job of task i.
Task τ_1: the jobs that completed before time 0 had length $C_{1,-3} = 7, C_{1,-2} = 6, C_{1,-1} = 8$.
Task τ_2: the jobs that completed before time 0 had length $C_{2,-3} = 18, C_{2,-2} = 13, C_{2,-1} = 17$.
Task τ_3: the jobs that completed before time 0 had length $C_{3,-3} = 2, C_{3,-2} = 7, C_{3,-1} = 3$.

(a) [6 points] Use the grid below to depict the schedule that is produced by the considered scheduler assuming that the future length of the next two jobs of each task is known and provided in Table 1.

Table 1: Arrival pattern and ground truth length of $j_{i,0}$ and $j_{i,1}$ for $i = \{1, 2, 3\}$.

	$j_{i,0}$ **Arrival**	$j_{i,0}$ **Length**	$j_{i,1}$ **Arrival**	$j_{i,1}$ **Length**
τ_1	0	8	21	5
τ_2	0	14	23	11
τ_3	6	5	26	7

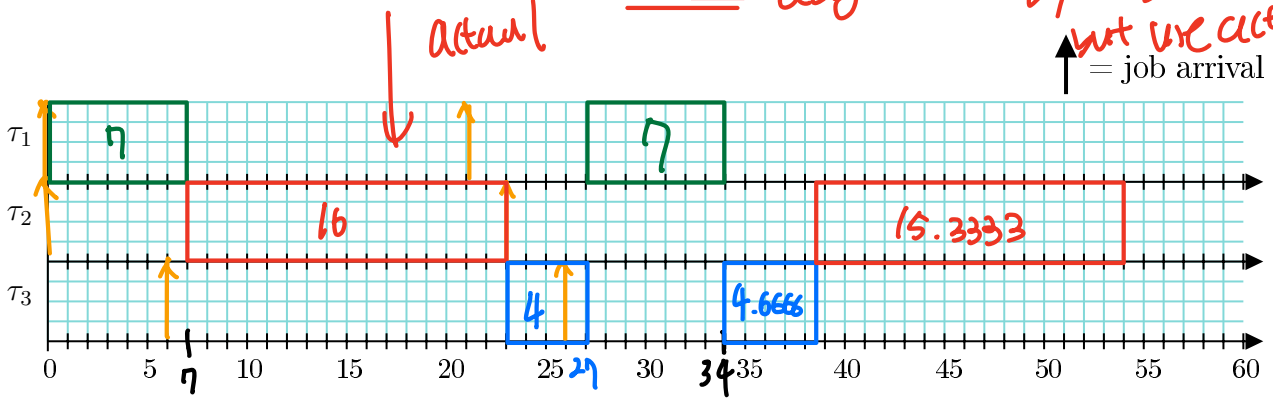


(b) [8 points] More realistically, consider the case where the length of jobs that will complete in the future is not known to the scheduler. Instead, it is estimated based on past job-length observations using a sliding window average with window size $w = 3$. Complete Table 2 with the predicted job lengths.

Table 2: Arrival pattern and length predictions for $j_{i,0}$ and $j_{i,1}$ for $i = \{1, 2, 3\}$.

	$j_{i,0}$ **Arrival**	$j_{i,0}$ **Pred. Length**	$j_{i,1}$ **Arrival**	$j_{i,1}$ **Pred. Length**
τ_1	0	7	21	7
τ_2	0	16	23	15.333
τ_3	6	4	26	4.666

(c) [8 points] Use the grid below to depict the schedule that is produced by the considered scheduler assuming that the future length of the next two jobs of each task is predicted according to the strategy considered in Part (b).



Problem 3

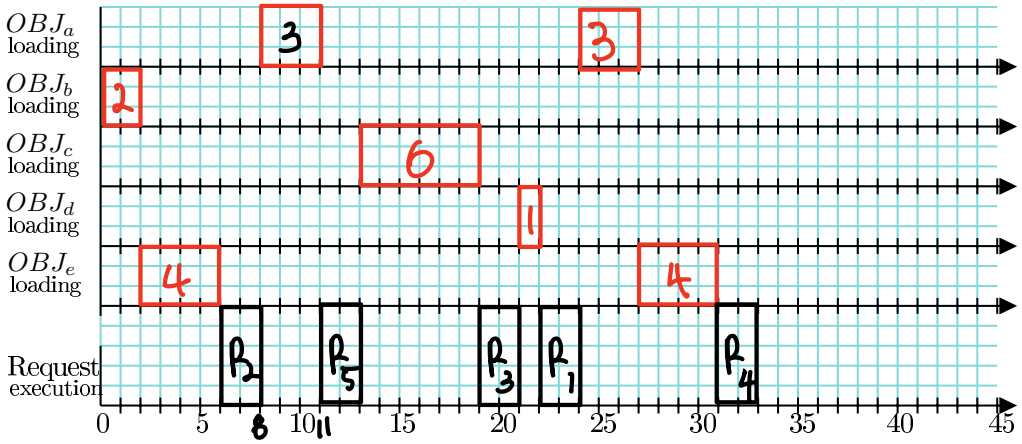
A system serves requests that perform computation over large memory objects. At time 0, there is a queue of 5 pending requests numbered R_1, R_2, R_3, R_4, R_5 in order of arrival, where R_1 is the oldest request. Each request requires some memory objects to be present in memory before it can be executed.

There are 5 objects: $OBJ_a, OBJ_b, OBJ_c, OBJ_d, OBJ_e$ and loading them into memory takes 3, 2, 6, 1, and 4 time units respectively. The requirements for the 5 queued requests are:
$R_1 = \{OBJ_c, OBJ_d\}, R_2 = \{OBJ_b, OBJ_e\},$
$R_3 = \{OBJ_b, OBJ_c\}, R_4 = \{OBJ_a, OBJ_c, OBJ_e\}, R_5 = \{OBJ_a, OBJ_b, OBJ_e\}.$

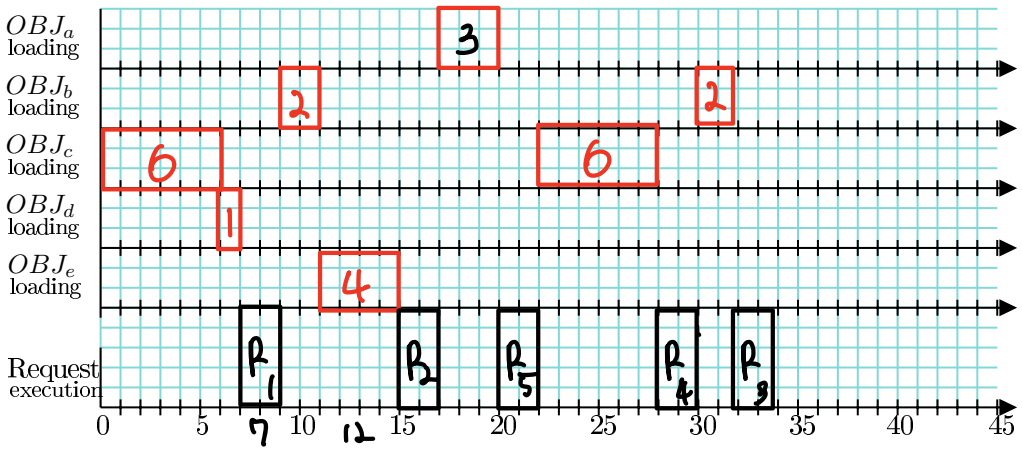
Once the requirements to execute a request are satisfied, **serving the request always takes 2 time units.** If some of the objects loaded in memory to serve the current request are also used by the immediate next request, **these objects can be kept in memory** and do not need to be re-loaded from disk. Conversely, **any object that was loaded and not used** by the immediate next request **is dropped from memory** and will have to be re-loaded as needed.

The system also abides to the following rules. (1) At time 0, no object is present in memory; (2) Loading multiple objects cannot be done in parallel, hence the time to load multiple objects is additive; (3) Loading any object for the next request to be served cannot be done until the current request has completed.

(a) **[6 points]** Formulate a notion of "scan length" that captures the time required to prepare the system to serve a request. Then, schedule the system using Shortest Scan Next (SSN) using the provided grid. *In the grid, use a separate axis to depict the time to load each object, and report on the bottom axis the time spent serving each request with a box labeled with the request ID.*



(b) [8 points] A pending request is considered *ready* if two or more objects are already in memory at a time when a scheduling decision is being computed. Use FR-FCFS to schedule the requests in the provided grid.

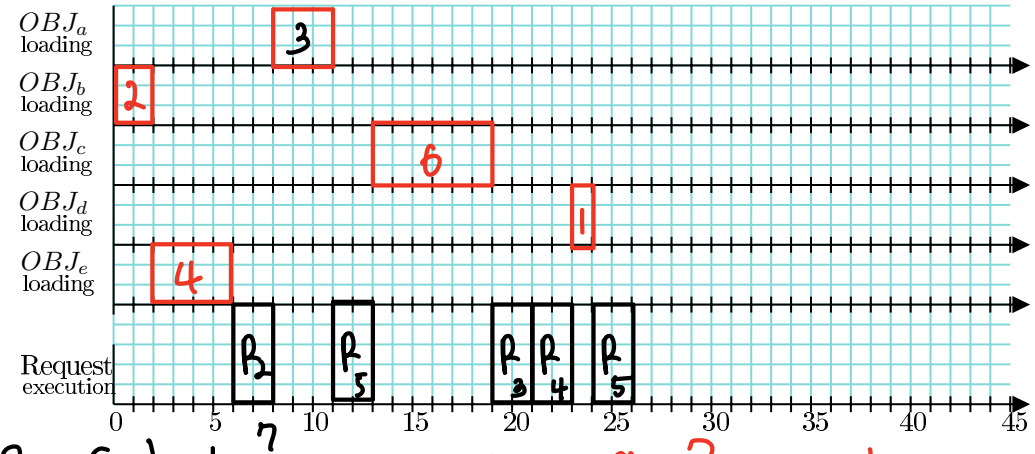


FR-FCFS → R_1

R_1 cd
R_2 be
R_3 bc b
R_4 ace b
R_5 a be

a 3 d 1
b 2 e 4
c 6

(c) [8 points] If the system could keep any object that is loaded in memory, in what order would you schedule the pending requests to minimize the average response time? For this question, simply assume that all the requests arrive at time 0 and that no object is in memory at time 0. Show the schedule in the grid below.



R_1 cd
R_2 be
R_3 bc
R_4 ace
R_5 abe

a 3 d 1
b 2 e 4
c 6

7, 12, 21, 23, 27

18

Problem 4

R-T Deadlines

Your team is engineering a controller for a safety-critical industrial assembly robot. In order to operate safely, 5 periodic tasks are implemented: (1) an obstacle detection task (OD), (2) an inertial measurement task (IM), (3) a path planning task (PL), (4) a joint constraint enforcement task (JC), and (5) an actuators update (AU) task.
The OD task is periodic at 100 Hz and has a WCET of 4 ms. The IM task has a period of 12 ms and a worst-case utilization of 50%. The PL task has a runtime between 1.5 ms and 3 ms and is released every 15 ms. The JC task has a WCET of 2 and a worst-case utilization of 25%. Lastly, the AU task is periodic with a 50 Hz rate and it runs for 6 ms in the worst case.

(a) [3 points] Complete Table 3 with the parameters of the tasks: WCET, period expressed in milliseconds, and utilization.

Task ID	WCET (ms)	Period (ms)	Utilization
OD	4ms	100Hz = 10ms	0.4
IM	6ms	12ms	0.5
PL	3ms	15ms	0.2
JC	2ms	8ms	0.25
AU	6ms	50Hz = 20ms	0.3

Table 3: Task Parameters.

(b) [5 points] Is the taskset schedulable on a single-processor CPU? Motivate your answer.

0.4+0.5+0.2+0.25+0.3 = 1.65

Not schedulable since it is bigger than 1

(c) [6 points] Is the taskset schedulable using EDF-FF on a two-processors CPU? Consider the tasks in the order in which they appear in the text and table.

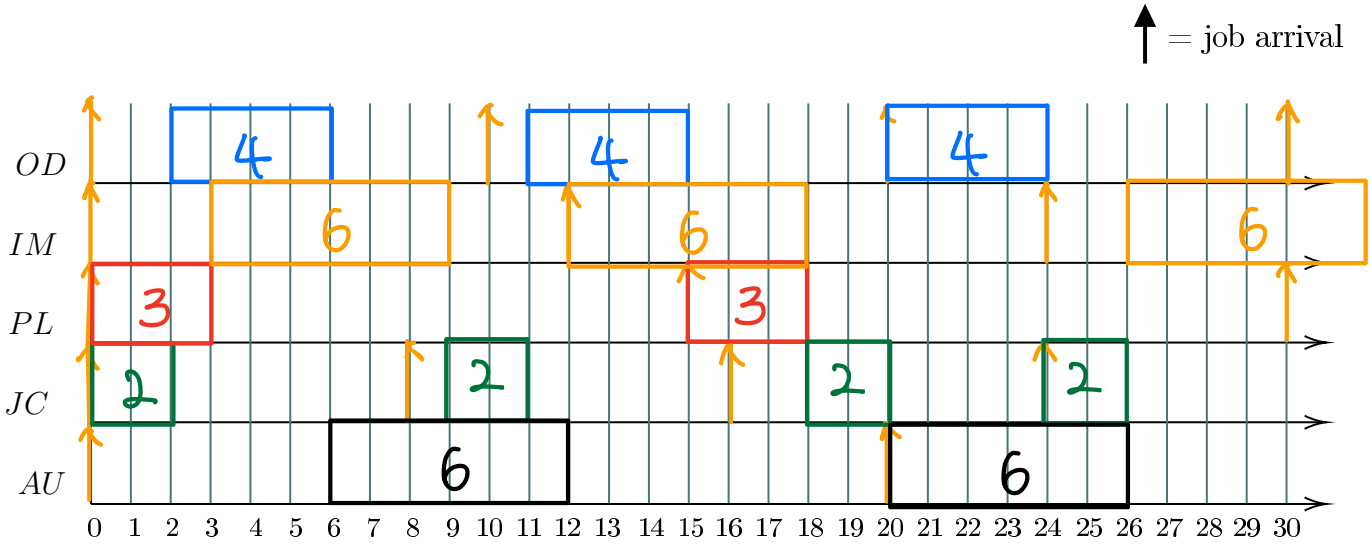
two processor

EDF: $\frac{B \cdot N+1}{N+1}$

$B = \lfloor \max \frac{C_i}{T_i} \rfloor = \frac{1}{0.5} = 2$

EDF: $\frac{2 \cdot 2+1}{2+1} = \frac{5}{3} = 1.6666$ schedulable

(d) [8 points] Using the grid below, produce the schedule that would be obtained when scheduling the provided taskset on a two-processor system using Global SJN up to time 30 ms. As usual, assume that the first release of each task is at time 0 ms. Break any tie by scheduling the task with the shorter period first.



Problem 5

Consider two processes P1 and P2 executing concurrently on a system. Reason on the possible interleaving of the two processes given their code. Note that both processes execute in an infinite loop and might use semaphores.

(a) [4 points] Consider the code of the two processes as provided in Listing 1 and 2 respectively. Assume that both processes are started at time 0 and that the processes are stopped after exactly three print statements have been executed in total by the two processes. What are all the possible outputs that can be produced? *Not all the blanks might need to be filled.*

```
1 Process P1:
2    do {
3      print("A ");
4      print("B ");
5    } forever;
```

Listing 1: Code of P1

```
1 Process P2:
2    do {
3      print("C ");
5    } forever;
```

Listing 2: Code of P2

ABC ✓ ACB ✓ CCC ✓ CAC ✓
CAB ✓ CCA ✓ ABA ✓ ACC ✓

(b) [8 points] Complete the code of the two processes in Listing 3 and 4 so that it can never happen that a "C" is printed between an "A" and a "B". For instance, the pattern `A B C C A B ...` is okay, while the pattern `A C B C A B ...` should not happen. For this purpose, use only one semaphore named `sem`. Make sure to provide the initialization value of `sem` at the beginning of Listing 3. *Not all the blanks might need to be filled.*

```
1  /* Shared Variables - START */
2
3  Semaphore sem = _____1_____;
4  /* Shared Variables - END */
5
6  Process P1:
7     do {
8
9        ___wait(sem)_____
10
11       ----------------------
12       print("A ");
13       print("B ");
14
15       ___signal(sem)_____
16
17       ----------------------
18    } forever;
```

Listing 3: Code of P1

```
6  Process P2:
7     do {
8
9        ___wait(sem)_____
10          .
11       ----------------------
12       print("C ");
13
14
15       ___signal(sem)_____
16
17       ----------------------
18    } forever;
```

Listing 4: Code of P2

(c) [8 points] Complete the code of the two processes in Listing 5 and 6 so that the only pattern that can be produced in output by the code has the following properties: (1) it starts with `A B` repeated TWICE; (2) it is followed by a single `C` ; (3) and then it repeats. I.e., it looks like: `A B A B C A B A B C ...`. Use two semaphores named `sem1` and `sem2`. Make sure to provide their initialization at the beginning of Listing 5. *Not all the blanks might need to be filled.*

```
1  /* Shared Variables - START */
2
3  Semaphore sem1 = ____2____;
4  Semaphore sem2 = ____0____;
5  /* Shared Variables - END */
6
7  Process P1:
8     do {
9        wait(sem1)
10    ----------------------
11
12       ----------------------
13       print("A ");
14       print("B ");
15       signal(sem2)
16    ----------------------
17
18       ----------------------
19    } forever;
```

Listing 5: Code of P1

```
7  Process P2:
8     do {
9        wait(sem2)
10    ----------------------
11       wait(sem2)
12       ----------------------
13       print("C ");
14
15       Signal(sem1)
16    ----------------------
17       Signal(sem1)
18       ----------------------
19    } forever;
```

Listing 6: Code of P2

[EXTRA BLANK PAGE (1)]

[EXTRA BLANK PAGE (2)]

Some Schedulability Tests

- Minimum Slowdown for job j_i at time t: $\frac{t-a_i+C_i}{C_i}$
- m tasks schedulable with RM if: $U \leq m(2^{1/m}-1)$
- Tasks schedulable with RM-FF on N CPUs if: $U \leq N(\sqrt{2}-1)$
- Tasks schedulable with EDF-FF on N CPUs if: $U \leq \frac{\beta N+1}{\beta+1}$ where $\beta = \lfloor \frac{1}{\max_k U_k} \rfloor$

Some Useful Numbers

- $2^{1/2} = 1.414213562$
- $2^{1/3} = 1.25992105$
- $2^{1/4} = 1.189207115$
- $2^{1/5} = 1.148698355$
- $2^{1/6} = 1.122462048$
- $2^{1/7} = 1.104089514$
- $2^{1/8} = 1.090507733$

