Midterm 2

Graded

Student

Jae Hong Lee

Total Points

107 / 110 pts

Question 1

Problem 1 - T/F

14 / 14 pts

(a)

✔ **+ 2 pts** (a) Correct

− 1 pt (a) Incorrect

+ 0 pts (a) Missing

(b)

✔ **+ 2 pts** (b) Correct

− 1 pt (b) Incorrect

+ 0 pts (b) Missing

(c)

✔ **+ 2 pts** (c) Correct

− 1 pt (c) Incorrect

+ 0 pts (c) Missing

(d)

✔ **+ 2 pts** (d) Correct

− 1 pt (d) Incorrect

+ 0 pts (d) Missing

(e)

✔ **+ 2 pts** (e) Correct

− 1 pt (e) Incorrect

+ 0 pts (e) Missing

(f)

✔ **+ 2 pts** (f) Correct

− 1 pt (f) Incorrect

+ 0 pts (f) Missing

(g)

✔ **+ 2 pts** (g) Correct

− 1 pt (g) Incorrect

+ 0 pts (g) Missing

(h)

✔ **+ 2 pts** (h) Correct

− 1 pt (h) Incorrect

+ 0 pts (h) Missing

(i)

✔ **+ 2 pts** (i) Correct

− 1 pt (i) Incorrect

+ 0 pts (i) Missing

(j)

✔ **+ 2 pts** (j) Correct

− 1 pt (j) Incorrect

+ 0 pts (j) Missing

Question 2

Problem 2 - Request Server Code **25** / 25 pts

2.1 ## (a) Scheduling Algorithm **8** / 8 pts

✔ **+ 8 pts** correct

+ 6 pts Round Robin

+ 2 pts Correct Quantum size

+ 3 pts Partial for mentioning SRT

+ 2 pts Partial credit for explanation for SRT

+ 0 pts incorrect / no sol

2.2 ## (b) Printout identification **9** / 9 pts

✔ **+ 9 pts** Correct identifies all 3 printouts + motivates answer

+ 6 pts 2 correct identified + motivation

+ 3 pts 1 correct + motivation

+ 4.5 pts Correctly identifies 3 no explanation

+ 3 pts Correctly identifies 2 no motivation

+ 1.5 pts correctly identifies 1 no motivation

+ 0 pts incorrect / no sol

+ 2 pts incorrectly identified Printout #1 as SJN

+ 1 pt Incorrectly identified Printout #1 but identified it was the one generated by the server implementation provided in Listing 1

2.3 ## (c) CDF identification **8** / 8 pts

+ 3 pts 1 correct + motivation

+ 6 pts 2 correct identified + motivation

✔ **+ 8 pts** Correct identifies all 3 graphs + motivates answer

+ 3 pts Correctly identifies 3 no explanation

+ 2 pts Correctly identifies 2 no motivation

+ 1 pt correctly identifies 1 no motivation

+ 0 pts incorrect / no sol

Question 3

Problem 3 - Real-time Scheduling **24** / 24 pts

3.1 (a) Utilizations **3** / 3 pts

- ✔ **+ 3 pts** Correct
- **+ 2 pts** 4/6 correct
- **+ 1 pt** 2/6 correct
- **+ 0 pts** incorrect / no sol

3.2 (b) RM-FF Scheduling **7** / 7 pts

- ✔ **+ 7 pts** Correct
- **+ 1 pt** Task 1 correct
- **+ 1 pt** Task 2 correct
- **+ 1 pt** Task 3 correct
- **+ 1 pt** Task 4 correct
- **+ 1 pt** Task 5 correct
- **+ 1 pt** Task 6 correct
- **+ 1 pt** correct conclusion NOT schedulable
- **+ 0 pts** incorrect / no sol
- **+ 5 pts** Task 1 to 5 is correct

3.3 (c) RM Schedulability **8** / 8 pts

- ✔ **+ 8 pts** Correct
- **+ 4 pts** Correct theoretical solution
- **+ 4 pts** Correct drawing of scheduling
- **+ 0 pts** incorrect / no sol
- **+ 1 pt** correctly determine the total utilization
- **+ 1 pt** correctly determine the RM schedulability bound
- **− 1 pt** incomplete graph but in generally trend is correct
- **+ 2 pts** Partially correct graph (that doesn't/minimally affects the conclusion)

3.4 **(d) EDF and WCET** **6** / 6 pts

- ✔ **+ 6 pts** Correct
- **+ 3 pts** correct setup for schedulability test
- **+ 1 pt** correct U_{other}
- **+ 0 pts** incorrect / no sol
- **+ 2 pts** correct numerical sol

Question 4

Problem 4 - Semaphores **24** / 24 pts

4.1 (a) Complete the code **7** / 7 pts

- **+ 2 pts** 1/7 lines correct
- **+ 3 pts** 2/7 lines correct
- **+ 4 pts** 3/7 lines correct
- **+ 5.5 pts** 4/7 lines correct
- **+ 6 pts** 5/7 lines correct
- **+ 6.5 pts** 6/7 lines correct
- ✔ **+ 7 pts** Correct
- **+ 0 pts** incorrect / no sol

4.2 (b) Semaphore initialization **10** / 10 pts

- ✔ **− 0 pts** Correct
- **− 10 pts** incorrect / no sol

incorrect initialization value

- **− 1 pt** 1 incorrect initialization value
- **− 2 pts** 2 incorrect initialization value
- **− 3 pts** 3 incorrect initialization value
- **− 4 pts** 4 incorrect initialization value
- **− 5 pts** 5 incorrect initialization value

incorrect semaphores

- **− 1 pt** 1 incorrect semephore
- **− 2 pts** 2 incorrect semephores
- **− 3 pts** 3 incorrect semephores
- **− 4 pts** 4 incorrect semephores
- **− 5 pts** 5 incorrect semephores

4.3 **(c) Behavior with dock = 2** **7** / 7 pts

✔ **+ 7 pts** Correct

+ 2 pts Correctly identify it won't work

+ 2 pts Correctly identifies a bug

+ 3 pts motivates answer

+ 0 pts incorrect / no sol

Question 5

Problem 5 - CFQ **20** / 23 pts

5.1 **(a) Initial state** **3** / 3 pts

- ✔ **+ 3 pts** Correct
- **+ 2 pts** 2/3 Task Queues correct
- **+ 1 pt** 1/3 Task Queues correct
- **+ 0 pts** incorrect / no sol

5.2 **(b) First qauntum** **3** / 3 pts

- ✔ **+ 3 pts** Correct
- **+ 1.5 pts** correct unordered queue
- **+ 1.5 pts** correct ordered queue
- **+ 0 pts** incorrect / no sol

5.3 **(c) Second quantum** **3** / 3 pts

- ✔ **+ 3 pts** Correct
- **+ 1.5 pts** correct unordered queue
- **+ 1.5 pts** correct ordered queue
- **+ 0 pts** incorrect / no sol

5.4 **(d) CFQ Optimization - ordering** **4** / 4 pts

- ✔ **+ 4 pts** Correct
- **+ 1 pt** Correct answer incorrect/guess explanation
- **+ 0 pts** incorrect / no sol
- **+ 3 pts** explanation

5.5 **(e) CFQ Optimization - head movement** **3** / 6 pts

- **+ 6 pts** Correct
- **+ 3 pts** Correct work for first variant
- **+ 3 pts** correct work for second variant
- ✔ **+ 3 pts** correctly identify improvement
- **+ 0 pts** incorrect / no sol

5.6 **(f) Reordering suffered** **4** / 4 pts

- ✔ **+ 4 pts** Correct
- **+ 2 pts** got 1 or 3 changes
- **+ 0 pts** incorrect / no sol

Department of Computer Science BOSTON UNIVERSITY

CS-350 - Fundamentals of Computing Systems

Midterm Exam #2
Fall 2024

Name: Jae Hong Lee

BU Username: jhonglee BU ID: U27565203

NOTE: *Please use only the provided space and the included extra page to answer the questions. If you do use the extra pages, make sure you reference them properly in your solutions. The very last page can be detached for your convenience.*

Problem #1:	/14
Problem #2:	/25
Problem #3:	/24
Problem #4:	/24
Problem #5:	/23
Total Score:	/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 a job that has a length shorter than the scheduler's quantum will never suffer preemption.	T
b.	It is possible that a taskset is schedulable on a multi-core system using Partitioned RM while it is not schedulable using Global EDF.	T
c.	Shortest Remaining Time (SRT) is a preemptive scheduling algorithm where starvation cannot occur by design. P	F
d.	In a DRAM controller, employing a FIFO scheduling policy allows to achieve higher memory throughput compared to when a FR-FCFS policy is used instead.	F
e.	When a `wait(sem)` operation is invoked on a semaphore `sem`, the calling process will always block.	F
f.	In EDF scheduling, a job belonging to a task with longer period might have higher priority compared to another job belonging to a task with shorter period.	T
g.	The specific policy of a work-conserving scheduler for a stateless resource might impact the average response time experienced by the workload.	T
h.	One can achieve the same level of coordination between threads if spinlocks are used instead of semaphores, but semaphore-based implementations waste less CPU cycles.	T
i.	Compared to SCAN, the C-SCAN disk scheduling algorithm reduces the worst-case response time for pending disk requests.	T
j.	It is NOT possible to implement a correct FIFO scheduler if the prediction of future job lengths has large estimation errors. X	F

Note: There are 10 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 14.

Problem 2

FIFO

A multi-threaded server uses multiple worker threads that process requests from a shared queue `the_queue`. Worker threads use the `add_to_queue(...)` and `get_from_queue(...)` procedure to add/fetch the requests to process to/from the shared queue in FIFO order. The code used by the worker thread to handle a request is provided in Listing 1. A parent thread (code omitted) is responsible for adding requests into the queue in FIFO order and initializing the `completed_exectime` to 0 upon request arrival.

```
struct timespec time_bit = {.tv_sec = 0, .tv_nsec = 10000000};
                                                   1000000 nsec
struct request {
   uint64_t req_id;                /* Integer request ID set by the client */
   struct timespec req_timestamp; /* time when request sent by client  */
   struct timespec req_length;    /* time length of the request */
};
struct request_meta {
   struct request request;
   struct timespec receipt_timestamp;    /* time when request enqueued */
   struct timespec start_timestamp;      /* time when request starts service */
   struct timespec completion_timestamp; /* time when request completed */
   struct timespec completed_exectime;.  /* partial amount of processing completed */
};
int worker_main (void * arg) {
  while (worker_done) {
    struct request_meta req;
    struct response resp;
    req = get_from_queue(the_queue);       get from Queue
    struct timespec compl_time = req.completed_exectime;       compl_time = 0
    /* First time processing this request -- mark start timestamp */
    if(compl_time.tv_sec == 0 && compl_time.tv_nsec == 0) {
      clock_gettime(CLOCK_MONOTONIC, &req.start_timestamp);     Start time Sm
    }
    timespec_add(&compl_time, &time_bit);
    if (timespec_cmp(&compl_time, &req.request.req_length) < 0) {
      busywait_timespec(time_bit);                       compl time smaller
      req.completed_exectime = compl_time;
      add_to_queue(req, the_queue);
    } else {
      struct timespec remainder = req.request.req_length;
      timespec_sub(&remainder, &req.completed_exectime);
      busywait_timespec(remainder);
      req.completed_exectime = req.request.req_length;
      clock_gettime(CLOCK_MONOTONIC, &req.completion_timestamp);
    }

    if (timespec_cmp(&req.completed_exectime, &req.request.req_length) == 0) {
      resp.req_id = req.request.req_id;
      resp.ack = RESP_COMPLETED;
      send(conn_socket, &resp, sizeof(struct response), 0);
      printout_completed_request(req);
      dump_queue_status(the_queue);
    }
  } return EXIT_SUCCESS;
}
```

Listing 1: Worker structure and dequeue operation

Recall that:

(1) `void timespec_add(struct timespec * a, struct timespec * b)` computes an addition of two timespec structures. The result of the addition `a + b` is stored in a.

(2) `void timespec_sub(struct timespec * a, struct timespec * b)` computes a subtraction of two timespec structures. The result of the subtraction `a - b` is stored in a.

(3) `int timespec_cmp(struct timespec * a, struct timespec * b)` compares two timespec structures. The function returns -1 if `a` is in the past compared to `b`; it returns 0 if `a` and `b` are the same; it returns 1 if `a` is in the future compared to `b`.

a bigger smaller return -1 0 1
same

(a) [8 points] What is the scheduling algorithm and its associated parameters (if any) employed by the server in the provided implementation? Motivate your answer.

The worker main scheduling algorithm is Round Robin Algorithm

Top of the code, there is, a shared and not changed timespec

"time_bit" each time worker compare the letted time bit

100000000 ns which is 10ms to the amount of work requested,

if the rest work is bigger than the time bit then it only

busy wait time bit and put it back to the queue.

so this is preemptive round robin scheduler.

RR
SJN FIFO

(b) **[9 points]** Take a look at the following printouts produced by three different server 3 diff implementations. In all three cases, the client sends three requests R0, R1, and R2 at time 0. The format is the following, where (1) req. ID is the request ID, (2) <sent ts>, (3) <receipt ts>, (4) <start ts>, and (5) <completion ts> are the timestamps at which the request was (2) sent by the client, (3) recevied by the parent thread, (4) started being worked on, and (5) completed by the worker thread.

```
R<req. ID>:<sent ts>,<req. length>,<receipt ts>,<start ts>,<completion ts>
                                      (3)          (4)          (5)
```

Printout #1:

```
T0 R1:0.000000,1.000000,0.000136,0.010166,3.010273
Q:[R2,R0]
T0 R2:0.000000,2.000000,0.000138,0.020167,5.010416
Q:[R0]
T0 R0:0.000000,3.000000,0.000134,0.000165,6.000512
Q:[]
```

Arrived at the same time

Printout #2:

```
T0 R1:0.000000,1.000000,0.000102,0.000148,1.000148
Q:[R0,R2]
T0 R2:0.000000,2.000000,0.000106,1.000203,3.000209
Q:[R0]
T0 R0:0.000000,3.000000,0.000100,3.000252,6.000253
Q:[]
```

Printout #3:

```
T0 R0:0.000000,3.000000,0.000090,0.000134,3.000136
Q:[R1,R2]
T0 R1:0.000000,1.000000,0.000093,3.000197,4.000198
Q:[R2]
T0 R2:0.000000,2.000000,0.000097,4.000258,6.000261
Q:[]
```

FIFO

Identify the policy used by the server implementations that correspond to each printout and specify which one is the printout that would be generated by the server implementation provided in Listing 1. Motivate your answer.

print out 1 has the three print at R1, R2, R3 when look at the start time. R0 is the first which got arried the server but it ended last, then R1 and R2 are followed. and when looked at it R2 is done earlier than R3 so I can see that the server rotates each jobs and finishing it by each quantum 1.000000 so printout 1 is RR which is Listing 1

Print out 2 is start timestamp is R1, R2, R0 so it is shortest job order so print out 2 is Shortest job next (SJN)

printout 3 is R0, R1, R2 order so it is First ~~one~~ first served (FIFO)

(c) [8 points] Consider the CDF of response times produced by three different server implementations on exactly the same input requests generated by the client. The three servers implement three different scheduling policies, namely FIFO, SJN, and HSN. Determine which policy is used in each plot corresponding to Figure 1, Figure 2, and Figure 3. Motivate your answer.

First of all Figure 2 is FIFO, since the CDF of Response time is gradually increased because FIFO does not consider job length or slow down It just takes each request so Figure 2 is FIFO.

Figure 1 and Figure 3, they both have a radical increase response time CDF plots. but Figure 1 has worst WCET around 60s Compare to Figure 3, WCET around 10s. HSN takes Highest slow down to consider the next time so considerable HSN has better WCET so Figure 1 is SJN. Figure 3 is HSN

WCET

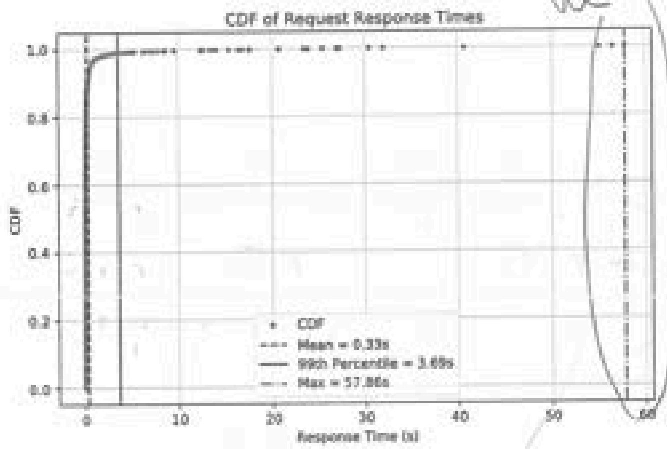


Figure 1: Response Time CDF Plot #1

SJN

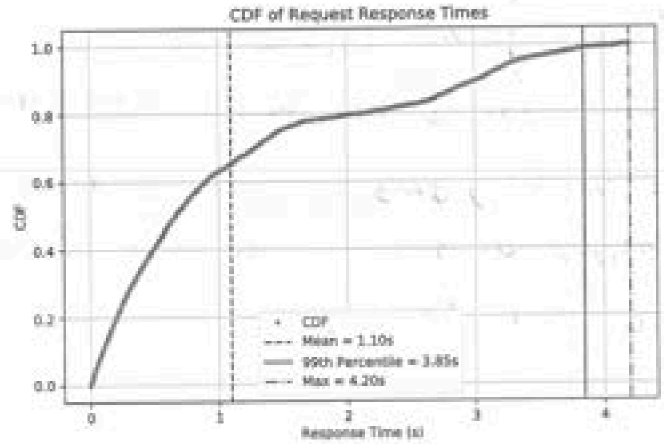
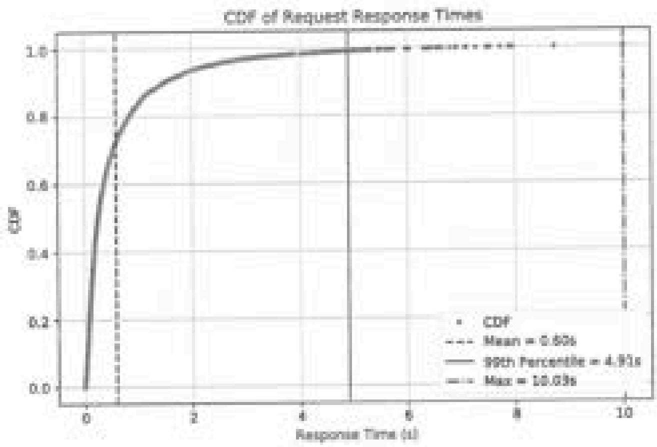


Figure 2: Response Time CDF Plot #2

FIFO



HSN.

Figure 3: Response Time CDF Plot #3

Problem 3

You are tasked with evaluating the schedulability of the following real-time tasks for an autonomous drone navigation system. The system consists of 6 periodic tasks with the parameters reported in Table 1. All the times are reported in milliseconds (ms).

Task	Function	WCET (C)	Period (T)	Utilization (U)
Task 1	Sensor Data Acquisition	5	15	0.333
✓ Task 2	Obstacle Detection & Avoidance	3	11	0.273
Task 3	Path Planning Update	2	6	0.333
✓ Task 4	Stabilization Control	4	9	0.444
✓ Task 5	Battery Monitoring	2	16	0.125
✓ Task 6	Telemetry to Ground Base	1	15	0.067

Table 1: Task Parameters for the Drone Navigation System

(a) **[3 points]** Compute the utilization for each task and fill in the last column of Table 1. Use the space below for any intermediate calculation.

Task 1 $= \frac{5}{15} = 0.3333$

Task 2 $= \frac{3}{11} = 0.2727... = 0.273$

Task 3 $= \frac{2}{6} = 0.333 = 0.333$

Task 4 $= \frac{4}{9} = 0.444 = 0.444$

Task 5 $= \frac{2}{16} = \frac{1}{8} = 0.125$

Task 6 $= \frac{1}{15} = 0.0666... = 0.067$

(b) **[7 points]** Determine if the system is schedulable on 2 processors using the Rate Monotonic First-Fit (RM-FF) partitioning algorithm. Include all intermediate calculations, final processor assignments, and utilization values to motivate your final answer.

There are 2 processor

so $N = 2$

The total utilization is

$U_1 + U_2 + U_3 + U_4 + U_5 + U_6 = 1.5746\ldots \approx 1.575$.

schedulability test for RM-FF

$= 2(\sqrt{2} - 1) = \underline{0.828}$

so RM-FF Test is $1.575 > 0.828$, in conclusion

	U
Task 1	0.331
Task 2	0.273
Task 3	0.331
Task 4	0.444
Task 5	0.125
Task 6	0.067

0.606 0.777 0.331 0.8444

P1	P2
Task 5	X
Task 2	Task 4
Task 1	Task 3

0.828

$m(2^{\frac{1}{m}} - 1)$

Task 1 + Task 2 = 0.606 < $2(2^{\frac{1}{2}} - 1) = 0.828$

Task 1 + Task 2 + Task 3 = 0.939 > $3(2^{\frac{1}{3}} - 1) = 0.779$ X

Task 1 + Task 2 + Task 4 = 1.05 > 0.779 X

Task 3 + Task 4 = 0.777 < 0.828 (O)

Task 1 + Task 2 + Task 5 = 0.731 < 0.779 (O)

Task 3 + Task 4 + Task 6 = 0.844 > 0.779

$\boxed{\text{Not schedulable}}$

(c) [8 points] Instead of using RM-FF, you decide to try and manually partition the tasks between the two processors. Focusing on processor 1, use any approach at your disposal to determine if tasks *T2*, *T4*, *T5*, and *T6* can be scheduled together using RM. You may use the grid provided below if necessary.

0.908

T_2	3	11
T_4	4	9
T_5	2	16
T_6	1	15

2 processor

Not schedulable

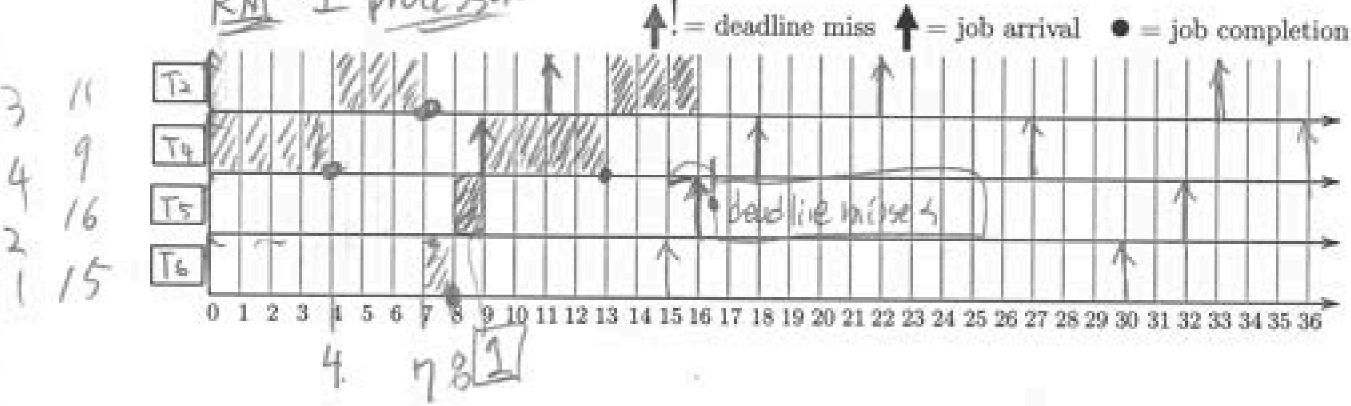
0.908 Global

0.90%

$4(2^{\frac{1}{4}}-1)$

0.7568

RM - 1 processor



(d) [6 points] Consider now 5 tasks, namely *T2*, *T3*, *T4*, *T5*, and *T6* that you want to schedule on a single processor using EDF. Your colleagues told you that they might be able to optimize the implementation of *T4* to improve its runtime. What is the maximum WCET that *T4* can have to ensure the schedulability of these 5 tasks under EDF?

1 cpu

$U_2 + U_3 + U_4 + U_5 + U_6 < 1$

$0.273 + 0.333 + \frac{x}{9} + 0.125 + 0.067 < 1$

$0.798 + \frac{x}{9} < 1$

$\frac{x}{9} < 0.202$

$x = 1.818$

$\boxed{\text{maximum WCET } 1.818\text{ms}}$

Problem 4

Alice is creating a ferry-boat company that does tours of the local wildlife of the Charles River. Each of her boats carries 20 passengers. Her fleet has a total of 3 boats.

A boat will attempt to dock at the station once it has completed a tour. Only one boat may be at the dock at a time. Then, when at the dock, the boat will allow all passengers to disembark. n= 1

Only once all passengers have disembarked will the boat accept new passengers. The boat will leave the station only once 20 new passengers have boarded. Then, the boat will do it's tour and, once completed, wait until it can dock again.

Alice needs to ensure that: The boat always rides with exactly 20 passengers; No passengers will jump off the boat while the boat is running; No passengers will jump on the boat while the boat is running; No passengers will request another boat ride before they can get off the boat.

(a) **[7 points]** Complete the pseudo-code reported below with appropriate statements. You should have at most one statement per blank; not all the blanks need to be used; you should not need to use more statements than the number of blanks. Any loop construct (e.g. for...loop, while...loop) takes two statements; an if...else...endif takes three statements, and so on. If the same variable is accessed by multiple processes, the variable is shared. ***Do not** introduce additional semaphores beyond what already mentioned in the code.*

```
/* Global, shared variables */
var boat_id;
bool empty[3] = true;

/* END of global, shared variables */

Customer: 20
  wait(admit_to_boat);
  var local_boat_id = boat_id;

  BOARD_BOAT();

  signal(boarded);

  /* Go on ride */
  wait(disembark[local_boat_id]);

  ____________________

  signal(done_disembark);
```

Listing 2: Customer Process

```
Boat i:
  repeat:
    wait(dock);
    boat_id = i;
    /* boat enters boarding station */
    if(!empty[i]):
      for(20):
        signal(disembark[i]);
        wait(done_disembark);
    for (20):
      signal(admit_to_boat);
      wait(boarded);
    empty[i] = false;
    signal(dock);
    DO_TOUR();
  forever
```

Listing 3: Boat Process

(b) [10 points] List all the semaphores used in the code and their initialization value.

semp admit_to_boat = 0;
semp boarded = 0;
semp dock = 1;
semp done_disembark = 0;
semp disembark = [0, 0, 0];

(c) [7 points] Alice wants to expand her buisness to have two docks, instead of one. She changes the above implementation by setting the initialization value of the **dock** semaphore to 2. Will her new implementation still work as expected? Motivate your answer.

No / It wont work, since the two docks but the Boat and customer codes of semaphore are still the same. the data race and unsynchronization could work so because of other unsynchronized semaphore it could lead to dead lock

Problem 5

You have 3 applications running on your system, each with their own disk request queue. The system is employing a Completely Fair Queuing (CFQ) scheduler with a quantum of 2 requests. Each application is reading from different sections of the disk. The disk has 16 total positions (0-15). The current head position is at 7. The reordering of the dispatch queue happens through C-SCAN where the disk head reaches 15 (even if there are no requests in 15) and after completing it, goes all the way down to 0.
Table 2 reports the three applications and 4 requests per application. For each request, the table reports the disk position targeted by the request. Requests are admitted into the dispatch queue following the order in which applications appear in the table.

0~15 QCFQ=2 7

	App	Req. 1 Target	Req. 2 Target	Req. 3 Target	Req. 4 Target
A	Web Browser	12	2	5	2
B	Terminal	7	8	5	4
C	Perplexity	5	9	2	1

Table 2: Disk Positions Targeted by Application Requests

(a) **[3 points]** Provide the initial state of the system by filling the content of the per-application queues at time 0, before any scheduling decision is taken. Provide your answer by completing the diagram provided in Figure 4.

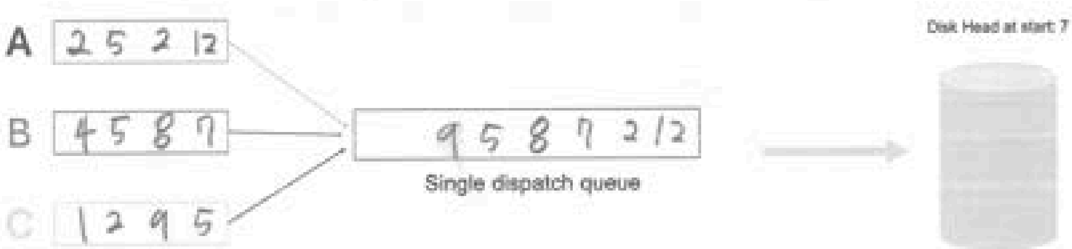


Figure 4: Initial system state.

(b) [3 points] Complete the diagrams provided in Figure 5 and 6 by showing the operation of the CFQ scheduler during the first quantum. Specifically, show status of the dispatch queue before reordering in Figure 5, and after the reordering in Figure 6.

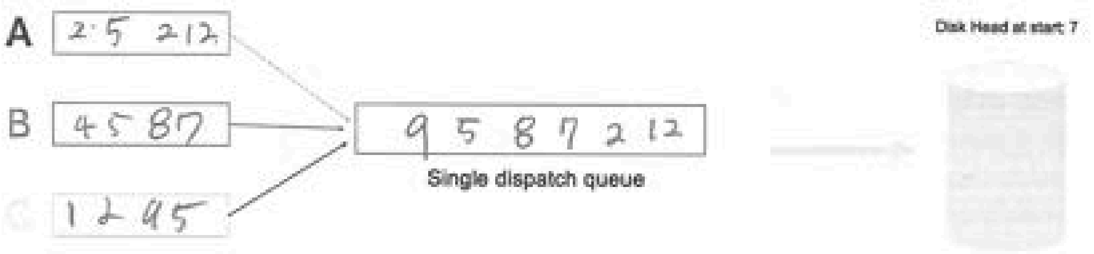


Figure 5: Unordered queue at first quantum.

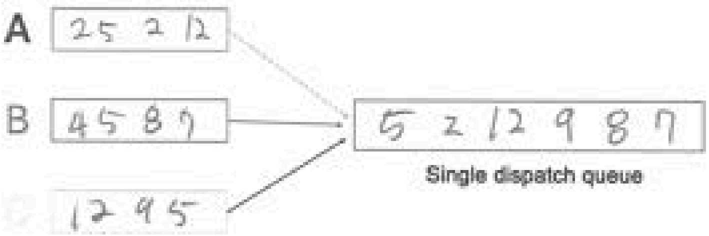


Figure 6: Ordered queue at first quantum.

(c) **[3 points]** Complete the diagrams provided in Figure 7 and 8 by showing the operation of the CFQ scheduler during the first quantum. First, fill in the position of the disk head at the beginning of the second quantum. Next, show the status of the dispatch queue before reordering in Figure 7, and after the reordering in Figure 8.

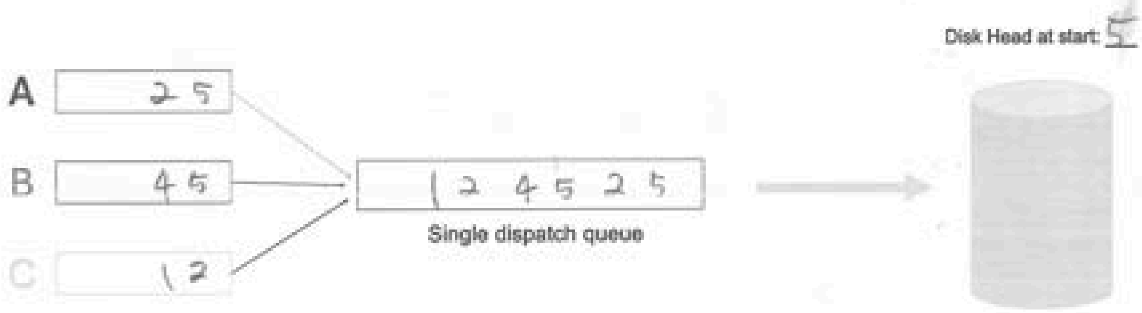


Figure 7: Unordered queue at second quantum.

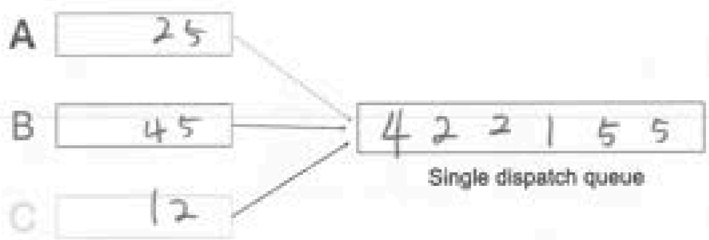


Figure 8: Ordered queue at second quantum.

(d) **[4 points]** Turns out, we have a pretty inefficient version of C-SCAN and we want to improve it. Your colleague in CS350 suggests to modify the current model to now do the following: (1) Once there are no more requests beyond the current position of the disk head, the disk position no longer goes all the way to 15. (2) Instead, it resets to the lowest position where there is an existing request at the time of checking for where to go next. For example, if the lowest index of request is at 2, the disk head will no longer go to 0, instead, go directly to 2.

Does this change the answer of reordering in part A? Explain why or why not.

Even The better version of C-SCAN comes in the reordering does not change. It would still be the same reordering just like part (B) but it will be efficient since no need to go all the way up or down of the disk.

(e) [6 points] Calculate the total disk movement using C-SCAN method of part A VS. the C-SCAN method defined in part B for the first quantum. Does the new proposed scheduler improve CFQ?

tie C SCAN method of part A

A) $\boxed{9\ 5\ 8\ 7\ \ 2\ 12}$ disk at 7

B) $\boxed{5\ 2\ 12\ 9\ 8\ 7}$

This is by using inefficient C-SCAN

A): 5+3+16+2+5+1+7+15+5+4 = $\boxed{63}$ movement

B): 0+1+1+3+3+15+2+3 = $\boxed{28}$ movement

It will be much faster with CFQ in part B

(f) [4 points] Consider the 4^{th} request of Application B 4 in Table 2. How many units of reordering will this request suffer due to the C-SCAN policy used by the CFQ scheduler? In other words, by how spots in the queue will this request be delayed compared to the unordered queue?

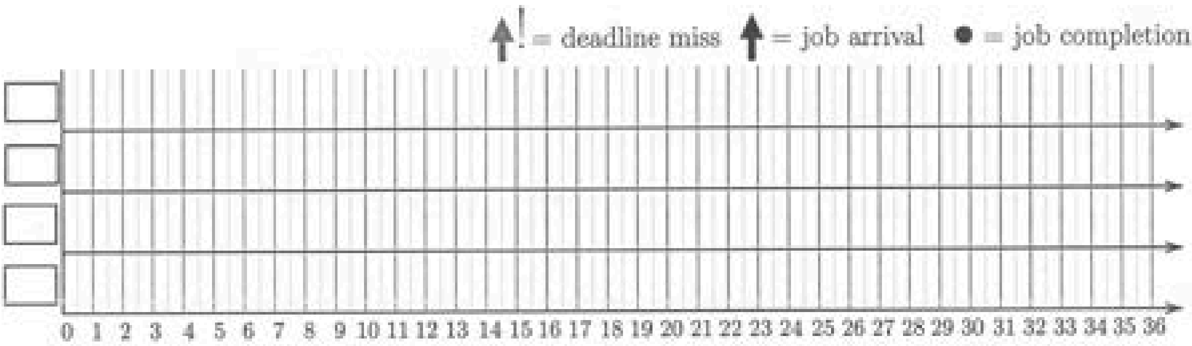
$\boxed{\text{2 units.}}$ It will be delayed to 2 units behind by reordering

12 4 5 2 5

to 2 units.

4 2 2 1 5 5

[EXTRA GRIDS & SPACE (*Use only if necessary and reference properly.*)]



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

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