United International University
Course: Operating Systems (CSE 4509), Summer 2025
Class Test 2, Set A
Total Marks: 20, Time: 40 minutes

Name		**ID**		**Section**	

1. Answer the following questions briefly and precisely. Each question contains 2 marks. **[5x2 = 10]**

a) In what kind of workload does STCF perform poorly, and why?

If a lot of process comes in decreasing order of duration time, then each time the new process is scheduled with context switch. Frequent context switch increase the turnaround time.

b) Does STCF guarantee fairness among processes? Why or why not?

No, STCF always chooses the process with shortest remaining time.

c) What is the main motivation behind using multiple queues in MLFQ scheduling?

To maintain different priority levels in order to balance responsiveness for short/interactive jobs and efficiency for long/CPU-bound jobs.

d) What problem does MLFQ try to solve that simple priority scheduling cannot?

MLFQ solves starvation and unfairness in fixed priority scheduling by dynamically adjusting priorities.

e) Is it necessary for threads in a process to have separate copies of the program Executable?

No, all threads of a process share the same program code as they're helpers of that program.

2. Consider The **Reader–Writer problem** that deals with synchronizing access to a shared resource (e.g., a database) where multiple readers can read simultaneously, but **writers need exclusive access**. In this version, if there are waiting readers and a writer, the **readers are given priority**.Multiple readers can read at the same time. A writer can only write when **no reader is reading** and **no new reader is waiting**. You are given with the pseudocode of the reader thread, you need to implement the **writer thread. [10 marks]**

```
mutex           // global lock
readCount = 0       // number of active readers
writeCount = 0      // 0 or 1, indicates writer is active
okToWrite          // condition variable for writers
okToRead           // condition variable for readers
```

Reader Thread	Writer Thread

`reader() {`
`  lock(mutex)`
 	
`  // Wait if a writer is active`
`  while (writeCount > 0) {`
`    wait(okToRead, mutex)`
`  }`
 	
`  readCount = readCount + 1`
`  unlock(mutex)`
 	
`  // ---- Critical Section ----`
`  readDatabase()`
 	
`  lock(mutex)`
`  readCount = readCount - 1`
 	
`  // If this was the last reader → allow writer`
`  if (readCount == 0) {`
`    signal(okToWrite)`
`  }`
 	
`  unlock(mutex)`
`}`
	`writer() {`
	
	`  lock(mutex);`
	
	`  while (readCount >0 || writeCount >0)`
	`  {   wait(okToWrite, mutex); }`
	
	`  writeCount = 1;`
	`  unlock(mutex);`
	`  writeDatabase();`
	
	`  lock(mutex);`
	
	`  writeCount = 0;`
	
	`  if (readCount >0)`
	
	`    signal(okToRead);`
	
	`  else`
	
	`    signal(okToWrite);`
	
	`  unlock(mutex);`
	`}`

United International University
Course: Operating Systems (CSE 4509), Summer 2025
Class Test 2, Set B
Total Marks: 20, Time: 40 minutes

Name		ID		Section	

1. Consider the following scheduling policy:

Dynamic Aging Priority Queue (DAPQ) Scheduling

- Like MLFQ, this policy uses multiple queues with different priorities.
- However, instead of fixed demotion or promotion rules, processes are assigned an **"aging score"** based on their wait time and CPU usage.
- The longer a process waits in the queue without getting CPU time, the faster its priority increases (aging).
- CPU-bound processes naturally sink toward lower queues, but if they wait too long, their aging score lifts them back up to avoid starvation.

Now answer the following questions:

a) How does the Dynamic Aging Priority Queue (DAPQ) differ from the traditional Multi-Level Feedback Queue (MLFQ) in preventing starvation? [3 marks]

DAPQ dynamically increases processes' priorities with aging as they wait for CPU share. As a result, a waiting process does not wait for a long and thus starvation is prevented.

b) If an I/O-bound process is continuously preempted by CPU-bound processes in MLFQ, how would DAPQ improve its response time? [3 marks]

In DAPQ, aging score boosts priorities quickly. So response time improves even if CPU bound jobs dominate.

c) *What could be a drawback of using dynamic aging scores for scheduling decisions compared to fixed promotion/demotion rules in MLFQ? [2 marks]*

The scheduler has to continuously update the aging score of the waiting processes which has an overhead.

d) In what kind of workload does STCF perform poorly, and why? [2 marks]

Solved in set A.

e) Is it necessary for threads in a process to have separate copies of the program executable?
[2 marks]

solved in set A.

2. A host of a party has invited N > 2 guests to his house. Due to fear of Covid-19 exposure, the [4] host does not wish to open the door of his house multiple times to let guests in. Instead, he wishes that all N guests, even though they may arrive at different times to his door, wait for each other and enter the house all at once. The host and guests are represented by threads in a multi threaded program. Given below is the pseudocode for the host thread, where the host waits for all guests to arrive, then calls openDoor(), and signals a condition variable once. You must write the corresponding code for the guest threads. The guests must wait for all N of them to arrive and for the host to open the door, and must call enterHouse() only after that. You must ensure that all N waiting guests enter the house after the door is opened. You must use only locks and condition variables for synchronization. The following variables are used in this solution: lock m, condition variables cv host and cv guest, and integer guest count (initialized to 0). You must not use any other variables in the guest for synchronization. [8 marks]

Host	Guest

```
mutex_t m;
cond_t cv_host, cv_guest;

void *host(void *args) {
    lock(m);
    while(guest_count < N) {
        wait(cv_host, m);
    }
    openDoor();
    signal(cv_guest);
    unlock(m);
}
```
	solved in the regular class.