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]

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]

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

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

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

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);`
`}`	