# CSC112 Spring 2025 Midterm Exam

Student Name: _______________ID： _______________

| Total Points | |
|---|---|

**Note: This exam paper should be kept confidential and any dissemination violates copyright.**

| Q1 | Q2 | Q3 | Q4 | Q5 |
|---|---|---|---|---|
| /20 | /20 | /10 | /20 | /30 |

**Q1 Multiple-choice questions: enter your answer keys here:**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| | | | | | | | | | |

**Q1. Multiple-choice. (20 pts)**
For the following multiple-choice questions, each question has exactly one correct answer key. If multiple choices are correct, choose the answer key "All of the above". **Fill in the answer keys in the table above. (Answer keys written in the question area will not be counted.)**

1. What is a process in an operating system?
A) A static program stored on disk
B) An active entity with a program counter and stack
C) A thread of execution
D) A section of memory

2. What is dual-mode operation in an operating system?
A) Running two operating systems simultaneously
B) Providing two modes: kernel mode and user mode
C) Allowing two users to access the same process
D) Switching between two CPUs dynamically

3. What does "protection" in an operating system ensure?
A) That processes cannot interfere with each other or the OS itself
B) That all applications run in kernel mode for efficiency
C) That users have unrestricted access to hardware resources
D) That only one application can run at a time on the machine

4. Which of the following is NOT typically included in a process control block (PCB)?
A) Process ID
B) Program counter

C) Source code
D) Open file descriptors

5. What is the purpose of the fork() system call?
A) To create a new thread
B) To make a duplicate copy of the calling process
C) To execute a new program
D) To terminate a process

6. What is the purpose of the wait() system call?
A) Puts the calling process to sleep
B) Waits for any child process to terminate
C) Waits for any parent process to terminate
D) Waits for available CPU time

7. What is the main difference between a process and a thread?
A) Processes are in the kernel space, and threads are in the user space
B) Threads cannot have multiple instances, but processes can
C) Processes have their own address spaces, while threads can share an address space
D) Threads are managed by the kernel, while processes are managed by the user

8. What prevents starvation in the ticket lock implementation?
A) Random backoff
B) FIFO queue based on ticket numbers
C) Priority inheritance
D) Timeout mechanisms

9. What ensures fairness in ticket locks?
A) Test-and-Set instruction
B) Fetch-and-Add atomic operation
C) Compare-and-Swap
D) Disabling interrupts

10. What happens when sem_wait() is called on a semaphore with value 1?
A) Decrements the value to 0 and does not block
B) Increments the value to 2 and does not block
C) Blocks until sem_post() is called
D) It does not modify the value and does not block

11. Which of the following is NOT a necessary condition for deadlock?
A) Mutual exclusion
B) Hold and wait
C) No preemption
D) Fair Scheduling

12. In a Resource-Allocation Graph (RAG), a deadlock is certain if:
A) There is a cycle and each resource has multiple instances
B) There is no cycle
C) There is a cycle and all resources have single instances
D) A thread requests two resources simultaneously

13. In the context of the Dining Philosophers problem, which solution can prevent deadlock?
A) Allowing each philosopher to pick up forks in any order
B) Requiring each philosopher to pick up both forks simultaneously in one atomic operation
C) Requiring each philosopher to pick up his left fork before his right fork

D) Removing all forks from the table

14. In the context of the two-armed lawyers problem, where there is a pile of chopsticks at the center of the table, which solution can prevent deadlock?
A) Allowing each lawyer to pick up chopsticks in any order
B) Requiring each lawyer to pick up both chopsticks simultaneously in one atomic operation
C) Requiring each lawyer to pick up his left chopstick before his right chopstick
D) Removing all chopsticks from the table

15. In the producer-consumer problem, why must the mutex be acquired after sem_wait(emptySlots)?
A) To prevent buffer overflow
B) To avoid deadlock
C) To ensure proper scheduling
D) To maintain thread priority

16. What is the main difference between spinlocks and semaphores?
A) Spinlocks use busy waiting, while semaphores allow threads to sleep
B) Spinlocks are faster in all scenarios
C) Semaphores can only be used for mutual exclusion
D) Spinlocks can only be used on single-core systems

17. What is the purpose of using a "while" loop instead of an "if" statement when checking a condition variable in a monitor?
A) To improve performance
B) To handle spurious wakeups
C) To reduce code complexity
D) To allow more threads to enter the critical section

18. Only in preemptive scheduling (not in non-preemptive scheduling), a process can transition directly from:
A) Running → Waiting
B) Running → Ready
C) Ready → Terminated
D) Waiting → Ready

19. Which of the following scheduling algorithm suffers from the Convoy effect, where short jobs are stuck behind long jobs?
A) Shortest Job First (SJF)
B) Shortest Remaining Time First (SRTF)
C) ) Round-Robin (RR)
D) Fixed-Priority (FP)

20. In Round Robin scheduling, if there are 10 jobs in the ready queue and time quantum=10ms, what's the maximum wait time for any job?
A) 40ms
B) 80ms
C) 90ms
D) 100ms

**Q2 Processes and Threads (20 pts)**
For these questions, assume there is no error, i.e., the return value of fork() or exec() is never negative.
a) (5 pts) What is the output of the program below? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
int ret = fork();
if(ret == 0) {
  exec(SOME_COMMAND); //SOME_COMMAND is a Linux command that does not print
anything
  printf("Child\n");
}
else {
  wait(NULL);
  printf("Parent\n")
}
```

ANS:

b) (5 pts) What is the output of the program below? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
int ret = fork();
if(ret==0) {
  printf("1");
  exec(SOME_COMMAND); //SOME_COMMAND is a Linux command that does not print
anything
} else {
  printf("2");
}
printf("3");
```

ANS:

c) (5 pts) What is the output of the program below? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
int ret = fork();
if(ret==0) {
  printf("1");
} else {
  printf("2");
}
printf("3");
```

ANS:

d) (5 pts) What is the output of the program below? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
int main() {
  print (“A”);
  int pid = fork();
  print (“B”);
  if (pid) wait(NULL);
  print (“C”);
}
```

ANS:

**Q3 Synchronization (10 pts)**
a) (5 pts) Consider the following concurrent program, where three threads access a shared variable x without mutex locks. What are the possible final values of x after all threads finish execution? Explain why.

```
int x=0; //x is a global shared variable

//Thread T1:
x = x + 1;

//Thread T2:
x = x - 1;

//Thread T3:
x = x * 2;
```

ANS:

b) (5 pts) Consider the following concurrent program, where three threads access a shared variable x within critical sections protected by mutex locks. What are the possible final values of x after all threads finish execution? Explain why.

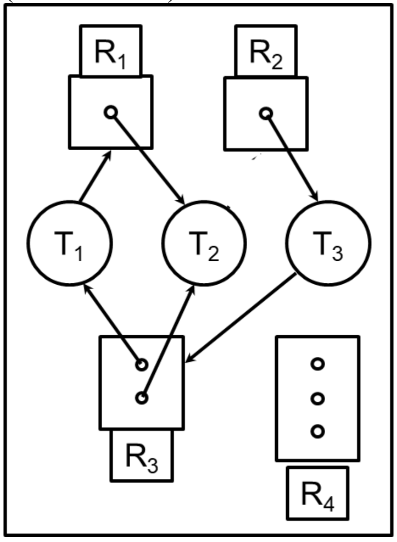
```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int x=0; //x is a global shared variable

//Thread T1:
pthread_mutex_lock(&mutex);
x = x + 1;
pthread_mutex_unlock(&mutex);

//Thread T2:
pthread_mutex_lock(&mutex);
x = x - 1;
pthread_mutex_unlock(&mutex);

//Thread T3:
pthread_mutex_lock(&mutex);
x = x * 2;
pthread_mutex_unlock(&mutex);
```

ANS:

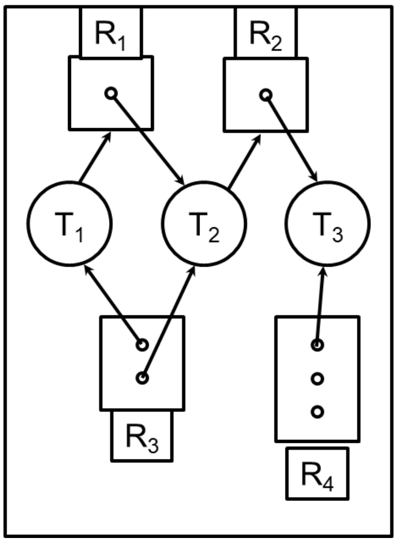
**Q4 Deadlocks (20 pts)**

Consider the following Resource Allocation Graph with 3 processes and 4 resource types. Number of small circles in the box of resource Rj indicates the number of instances of resource Rj. An arrow from process Ti to resource Rj indicates that Ti requests 1 instance of Rj; an arrow from resource Rj to process Ti indicates that Ti is holding 1 instance of Rj. Run Banker's algorithm to check if the current state is safe, by writing out the matrices Max, Allocation and Need, and vectors Total and Available. If yes, give a safe sequence of process completions and fill in the table with the sequence of process completions without deadlock, and available resources after the completion of each process.

(Two variants)



OR



ANS:

**Q5 Scheduling (30 pts)**

a) (10 pts) Consider the sequence of processes with CPU burst time in parentheses: P1(10ms), P2(2ms), P3(2ms) arriving at time 0 in the order of P1, P2, P3. Calculate the average response time under 1) First Come, First Served (FCFS). 2) Shortest Job First (SJF). 3) Shortest Remaining Time First (SRTF). 4) Round-Robin (RR) with time quantum 2. 5) Fixed-Priority scheduling with the priority ordering P3>P2>P1. (There is no need to draw the Gantt chart, but please show the response time of each process R1, R2, R3 and calculate R=(R1+R2+R3)/3.)

ANS:

b) (20 pts) Consider the set of 2 processes whose arrival time and CPU/IO burst times are given below. For each scheduling algorithm (FCFS, SJF, SRTF, RR, Fixed-Priority (FP)), draw the Gantt chart by filling in the table with the PID that runs in each time slot, and calculate the response time for each process, and the average response time. **For RR scheduling, the time quantum is 1. For FP scheduling, assign P2 (PID 2) higher priority than P1 (PID 1)**. If a time slot is idle with no active process executing, then fill in X. (Except for any possible idle time slots at the end of schedule, leave them empty and do not fill in X.)
(Two variants)

| PID | Arriv. time | CPU Burst | IO Burst | CPU Burst | FCFS Resp. Time | SJF Resp. Time | SRTF Resp. Time | RR Resp. Time | FP Resp. Time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | 2 | 4 | | | | | |
| 2 | 1 | 1 | 2 | 2 | | | | | |
| | | | | | Avg RT | Avg RT | Avg RT | Avg RT | Avg RT |

OR

| PID | Arriv. time | CPU Burst | IO Burst | CPU Burst | FCFS Resp. Time | SJF Resp. Time | SRTF Resp. Time | RR Resp. Time | FP Resp. Time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | 2 | 4 | | | | | |
| 2 | 1 | 1 | 3 | 3 | | | | | |
| | | | | | Avg RT | Avg RT | Avg RT | Avg RT | Avg RT |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FCFS | | | | | | | | | | | | |
| SJF | | | | | | | | | | | | |
| SRTF | | | | | | | | | | | | |
| RR | | | | | | | | | | | | |
| FP | | | | | | | | | | | | |

Time 0 1 2 3 4 5 6 7 8 9 10 11 12

Gantt Chart