CSC256 Advanced OS Fall 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	/30	/20

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 option "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 the "kernel" in an operating system?
A) The hardware component managing memory
B) The one program running at all times on a computer
C) A user interface for applications
D) A type of application program

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. Which system call replaces the current process image with a new one?
a) `fork()`
b) `wait()`
c) `exec()`
d) `exit()`

4. What is the purpose of the `wait()` system call?
a) To create a new process

b) To wait for I/O operations to complete
c) To suspend the parent process until a child process terminates
d) To destroy a process

5. What is the difference between `wait()` and `waitpid()`?
a) `waitpid()` allows specifying which child process to wait for, while `wait()` does not.
b) `wait()` waits for all processes, while `waitpid()` waits only for threads.
c) Both are identical in functionality.
d) `waitpid()` suspends processes, while `wait()` terminates them.

6. In which state is a process when it is waiting for an I/O operation to complete?
a) READY
b) RUNNING
c) BLOCKED
d) TERMINATED

7. What is the primary advantage of threads over processes?
a) Threads have separate address spaces.
b) Threads are cheaper to create and manage than processes.
c) Threads cannot share resources like processes can.
d) Threads only exist in kernel mode.

8. What does multithreading allow in modern operating systems?
a) Multiple address spaces per thread
b) Concurrent execution within the same address space
c) Execution of only one thread at any time in a system
d) Elimination of kernel threads

9. In the Readers/Writers problem, why might writers starve?
A) Writers have higher priority
B) New readers continuously acquire the lock before writers
C) Semaphores are initialized incorrectly
D) Mutex locks are not used

10. Which synchronization primitive combines a mutex with condition variables?
A) Spinlock
B) Semaphore
C) Monitor
D) Ticket lock

11. What happens when sem_wait() is called on a semaphore with value 0?
A) Returns immediately
B) Increments the value to +1
C) Blocks until sem_post() is called
D) Causes a segmentation fault

12. Which condition variable operation wakes all waiting threads?
A) pthread_cond_signal()
B) pthread_cond_broadcast()
C) pthread_cond_wait()

D) pthread_cond_init()

13. A counting semaphore initialized to N allows:
A) Only one thread to access a resource
B) Up to N threads to access a resource simultaneously
C) Threads to bypass mutex locks
D) Priority inversion to occur

14. Which condition is NOT a necessary condition for deadlock?
A) Mutual exclusion
B) Hold-and-wait
C) Starvation
D) Circular wait

15. 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

16. In a Resource Allocation Graph (RAG) with a cycle and multi-instance resources:
A) Deadlock is certain
B) Deadlock is impossible
C) Deadlock is possible but not certain
D) Starvation must occur

17. 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

18. Which scheduling algorithm requires prior knowledge of job execution times?
A) FCFS
B) RR
C) SJF
D) Multilevel Queue

19. In exponential averaging for burst prediction ($\tau_n = \alpha t_{n-1} + (1-\alpha)\tau_{n-1}$), what does $\alpha=1$ imply?
A) Only consider historical average
B) Only consider most recent burst
C) Equal weight to all bursts
D) No prediction capability

20. What percentage of CPU time is lost to context switching if quantum=100 ms and switch cost=1 ms?
A) 1%
B) 2%

C) 5%
D) 10%

Q2 Processes and Threads (20 pts)

For these questions, assume there is no error, i.e., the return value of fork() is never negative. Assume round-robin scheduling algorithm like in Windows or Linux. (You need to provide the possible outputs and explain why. You do not need to draw the figures to show the parent child relationships.)

(a) (4 pts) What is the output of this program? If there may be multiple possible outputs, list ALL possible outputs, and explain why. Assume the virtual memory address of variable a in the initial process, &a = 0x12345678.

```
1 int main(void) {
2 int a = 1;
3 pid_t fork_ret = fork();
4 if (fork_ret > 0) {
5   a++;
6   fprintf(stdout, "Parent: int a is %d at %p \n", a, &a);
7 } else if (fork_ret == 0) {
8   a--;
9   fprintf(stdout, "Child: int a is %d at %p \n", a, &a);
10 } else {
11   printf("Fork error");
12 }
13 }
```

ANS:

(b) (4 pts) What is the output of this program? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
1 int main(void) {
2 char** argv = (char**) malloc(3 * sizeof(char*));
3 argv[0] = "/bin/ls";
4 argv[1] = ".";
5 argv[2] = NULL;
6 for (int i = 0; i < 10; i++) {
7   printf("%d\n", i);
8   if (i == 3) {
9       execv("/bin/ls", argv);
10  }
11 }
12 return 0;
13 }
```

ANS:

(c) (4 pts) What is the output of this program? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
1 int main(void) {
2 char** argv = (char**) malloc(3 *  sizeof(char*));
3 argv[0] = "/bin/ls";
4 argv[1] = ".";
5 argv[2] = NULL;
6 for (int i = 0; i < 10; i++) {
7  printf("%d\n", i);
8  if (i == 3) {
9      if(fork() == 0)
10         execv("/bin/ls", argv);
11 }
12 return 0;
13 }
```

ANS:

(d) (4 pts) What is the output of this program? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
1 int main(void) {
2 char** argv = (char**) malloc(3 *  sizeof(char*));
3 argv[0] = "/bin/ls";
4 argv[1] = ".";
5 argv[2] = NULL;
6 for (int i = 0; i < 10; i++) {
7  printf("%d\n", i);
8  if (i == 3) {
9      if(fork() > 0)
10         execv("/bin/ls", argv);
11 }
12 return 0;
13 }
```

ANS:

(e) (4 pts) What is the output of this program? If there may be multiple possible outputs, list ALL possible outputs, and explain why.

```
1 int main(void) {
2 char** argv = (char**) malloc(3 *  sizeof(char*));
3 argv[0] = "/bin/ls";
4 argv[1] = ".";
5 argv[2] = NULL;
6 for (int i = 0; i < 10; i++) {
7  printf("%d\n", i);
8  if (i == 3) {
9      ret = fork();
10     if(ret == 0)
11         execv("/bin/ls", argv);
12     if(ret > 0)
13         wait(NULL);
14 }
12 return 0;
13 }
```

ANS:

Q3 Synchronization (10 pts)

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

```
//Thread T1:
pthread_mutex_lock(&mutex);
x = x + 2;
pthread_mutex_unlock(&mutex);
```

```
//Thread T2:
pthread_mutex_lock(&mutex);
x = x - 2;
pthread_mutex_unlock(&mutex);
```

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

ANS:

(b) (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 + 2;
```

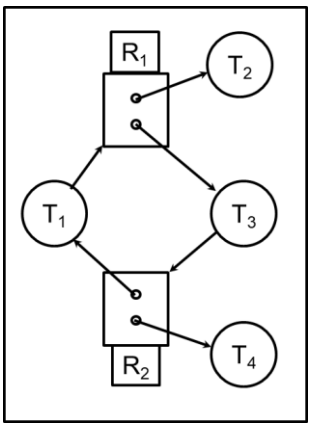
```
//Thread T2:
x = x - 2;
```

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

ANS:

Q4 Deadlocks (30 pts) Consider the following Resource Allocation Graphs (a) and (b). Use Banker's algorithm to check if it is possible for the system to be deadlocked. If no, give a safe sequence of process completions without deadlock. If yes, show an unsafe state that will result in a deadlock. (You need to give the Max, Allocation, Need matrices, Total and Available vectors, and Available resources after completion of each process.)

(a) (10 pts) Resource Allocation Graph 1 and corresponding tables.



Max

	R1	R2
T1		
T2		
T3		
T4		

Allocation

	R1	R2
T1		
T2		
T3		
T4		

Need

	R1	R2
T1		
T2		
T3		
T4		

Total

R1	R2

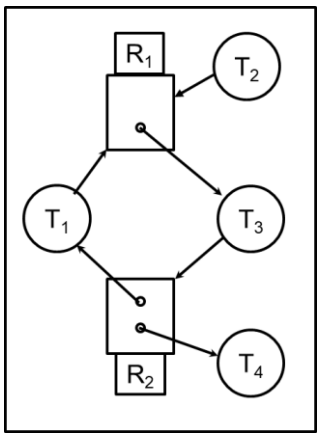
Available

R1	R2

Available

	R1	R2
Init	0	0

(b) (10 pts) Resource Allocation Graph 2 and corresponding tables.



Max

	R1	R2
T1		
T2		
T3		
T4		

Allocation

	R1	R2
T1		
T2		
T3		
T4		

Need

	R1	R2
T1		
T2		
T3		
T4		

Total

R1	R2

Available

R1	R2

Available

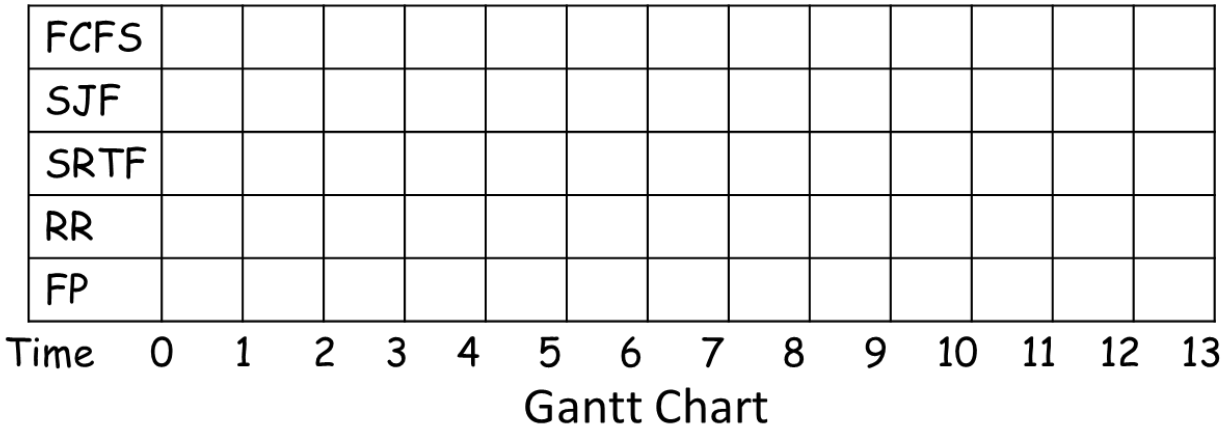
	R1	R2
Init	0	0

(c) (10 pts) Consider the Dining Lawyers problem. There are 3 lawyers P1 to P3, each with a different number of arms. P1 has 1 arm and needs 1 fork to eat; P2 has 2 arms and needs 2 forks to eat; P3 has 3 arms and needs 3 forks to eat. There is a pile of 3 forks at center of the table. Each lawyer picks up one fork at a time, and when he gets enough forks, he eats and then puts down all his forks. Use Banker's algorithm to check if it is possible for the system to be deadlocked. If no, give a safe sequence of process completions without deadlock. If yes, show an unsafe state that will result in a deadlock. (You need to give the Max, Allocation, Need matrices, Total and Available vectors, and Available resources after completion of each process.)

Q5 Scheduling (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. Assume that context switchoverhead is 0. For RR scheduling, the time quantum is 1. For RR, assume that an arriving process is scheduled to run at the beginning of its arrival time, i.e., it is added to the head ofthe queue upon arrival. **For FP scheduling, assign P2 (PID 2) higher priority than P1 (PID 1).** In case of a tie, prefer the running process to the newly arrived process, and if everything else is equal, prefer the process with smaller index. 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.)

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	5	3	2					
2	3	1	2	3					
					Avg RT	Avg RT	Avg RT	Avg RT	Avg RT



Gantt Chart