CS 3113 Introduction to Operating System - Fall 2025 – Exam 2

Tuesday, November 4th, 2025

Exam Instructions

- The exam is **closed to electronics** but **open to unlimited handwritten notes**.
- Printed slides may be used **only if each page contains handwritten notes**. Pages without handwritten notes are not permitted.
- You may use a **scientific calculator**; however, **mobile phone calculators are not allowed**.
- You will have **75 minutes** to complete the exam of 7 questions.
- Be sure to write your **full name (exactly as shown on the Canvas roster)** and your **OU ID** on your exam.

All the best!

Name:

Chih-Yu Chu

OU ID:

113595150

Total Score

Write your answers only in the provided boxes.

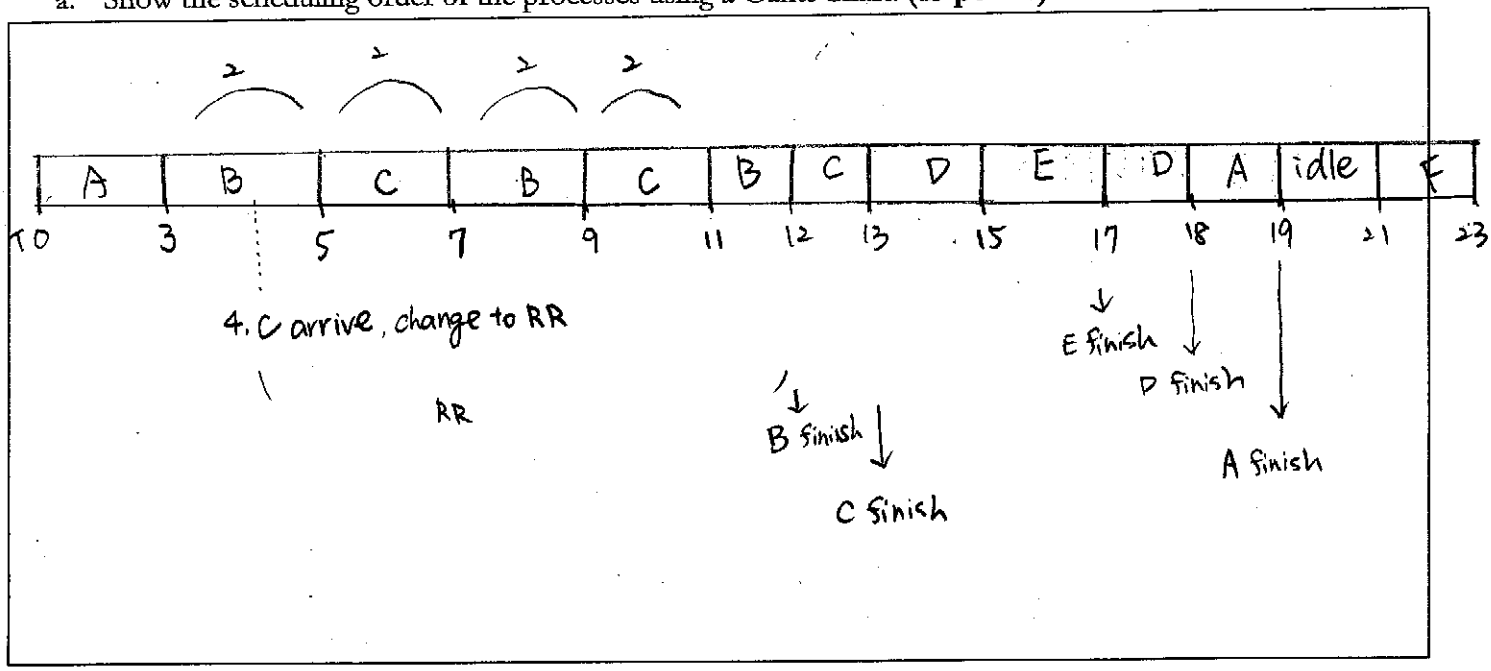
1. The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm.

Process	Priority	Burst Time	Arrival	Wait	Turn Around
A	1	~~4~~ 1	0	15	19
~~B~~	~~3~~	~~5 3 X~~	3	4	9
~~C~~	~~3~~	~~5 3 X~~	4	4	9
D	2	~~3~~ X	12	1+2=3	6
E	5	~~2~~	15	0	2
F	4	2	21	0	2

check: Burst + Wait = Turn

Each process is assigned a numerical priority, with **a higher number indicating a higher relative priority**. The scheduler will execute the highest-priority process first. For processes with the same priority, a round-robin scheduler will be used with **a time quantum of 2 units**. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue. The CPU can be idle whenever the system has no other available processes to run.

a. Show the scheduling order of the processes using a Gantt Chart. **(15 points)**



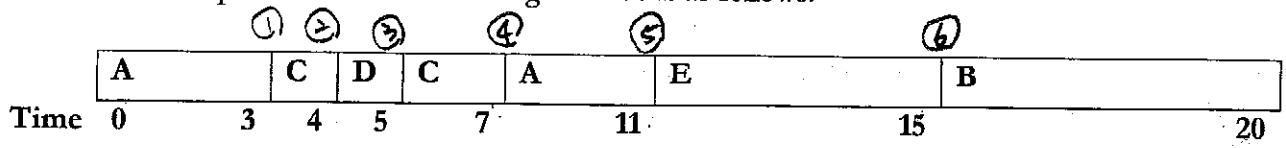
b. Determine the CPU utilization rate. **(5 points)**

$$\text{CPU utilization} = \frac{\text{total} - \text{idle}}{\text{total}} = \frac{23-2}{23} = \frac{21}{23}$$

2. The following processes are being scheduled using Shortest Remaining Time First (SRT).

Process	Burst Time	Arrival Time
A	7	0
B	5	3
C	3	3
D	1	4
E	4	5

The Gantt Chart of the processes scheduled using SRT looks as follows:



Σ time in ready queue

a. For each process, indicate its waiting time and turnaround time **(10 points)**

Finish - Arrival

Process	Waiting Time	Turnaround Time
A	4	11
B	12	17
C	1	4
D	0	1
E	6	10

b. What is the total number of context switches when CPU scheduled the above processes? Explain your answer. **(5 points)**

In Shortest-Remaining-Time-First Algorithm, context switch happen when one process change to others, either be preempted by other process or finish burst time and switch to other process.
So there are 6 context switches here.

3. Two threads: T1 and T2 are running concurrently. Thread 1 and Thread 2 share a global variable y. Initially y = 10, and its value is stored at memory address M1000.

Thread 1 executes: y = y + 5
Thread 2 executes: y = y - 3

The instructions related to the statement y = y + 5 are as follows:
1. LOAD R1, M1000
2. MOV R2, #5
3. ADD R3, R1, R2
4. STORE R3, M1000

The instructions related to the statement y = y - 3 are as follows:
1. LOAD R1, M1000
2. MOV R2, #3
3. SUB R3, R1, R2
4. STORE R3, M1000

Consider the following concurrent execution of Thread 1 and Thread 2 instructions in the CPU:

Time	Thread 1	Thread 2	R1	R2	R3	M1000
1		LOAD R1, M1000	10			10
2		MOV R2, #3		10		
3	LOAD R1, M1000		10			
4	MOV R2, #5			10		
5	ADD R3, R1, R2		15	15	5	
6	STORE R3, M1000					5
7		SUB R3, R1, R2	12	12	2	
8		STORE R3, M1000				2

10 points:

a. What is the value of R1 right after **LOAD R1 M1000** is executed at time 1?

10

b. What is the value of M1000 right after **MOV R2, #5** is executed at time 4?

10

c. What is the value of R3 right after **ADD R3, R1, R2** is executed at time 5?

R3 = _ + 5 = 5

5

d. What is the value of R3 right after **SUB R3, R1, R2** is executed at time 7?

R3 = 5 - 3 = 2

2

e. What is the value of M1000 right after **STORE R3, M1000** is executed at time 8?

store R3 = 2 to M1000

2

4. There is a global variable called totalItems. This variable represents the number of items in a shared inventory system. Multiple threads may concurrently call the **addItem()** and **removeItem()** functions:

```
int addItem(int quantity) {
    if (totalItems + quantity <= MAX_STOCK) {
        totalItems += quantity;
    }
    return totalItems;
}

int removeItem(int quantity) {
    if (quantity <= totalItems) {
        totalItems -= quantity;
    }
    return totalItems;
}
```

CS

CS

There is a race condition on the shared variable **itemCount**.

a. Identify the critical section on the **addItem** function. **(7 points)**

```
if (totalItems + quantity ≤ MAX_STOCK) {
    totalItems += quantity;
}
```

b. Rewrite the **removeItem** function and add a mutex lock with the acquire() and release() operations to avoid the race condition? **(8 points)**

```
acquire();
// CS
release();
// remain
```

```
int removeItem (int quantity) {
    acquire();
    if (quantity <= totalItems) {
        totalItems -= quantity;
    }
    release();
    return totalItems;
}
```

5. Six processes (A, B, C, D, E, and F) are competing to use a computer resource that has a maximum of 4 instances. The operating system synchronizes access using a non-busy-waiting semaphore S with the following operations: → S initial value is 4 , S=4.

```
wait(semaphore *S) {
    S->value--;
    if (S->value < 0) {
     add this process to S->list;
     block();
    }
}

signal(semaphore *S) {
    S->value++;
    if (S->value <= 0) {
     remove a process P from S->list;
     wakeup(P);
    }
}
```

10 points:

a. At time *i*, four instances of the resource are used each by A, B, E, and F.

What is the value of **S->value** time *i*? S = 4-4 = 0 A B E F

0

b. At time *i+1*, F releases the instance.

What is the value of **S->value** at time *i+1*? S = 0 +1 = 1 A B E

1

c. At time *i+2*, A releases the instance. What is the value of **S->value** at time *i+2*? B E

S = 1+1 = 2

2

d. At time *i+3*, C needs an instance of the resource and so does D. B E C D

What is the value of **S->value** at time *i+3*?

S = 2 - 2 = 0

0

e. At time *i+4*, F needs an instance of the resource.

What is the content of **S->list** at time *i+4*? S = 0 -1 = -1

F

6. A system is using a deadlock avoidance algorithm (Banker's algorithm). The snapshot of the system's current state is as follows:

Total resources			
A	B	C	D
3	14	12	12

a) Use the table below to trace the algorithm. Fill in the **Need** matrix, record the sequence in which processes receive resources, and update the **Work** vector whenever a process is allocated resources. In the bottom row, show the available resources after the final process receives its allocation. **(15 points)**

(Need = Max - Allocation)

Process	Allocation	Max	Need	Work = Available	**Sequence of Resource Allocation**	
	A B C D	**A B C D**	**A B C D**	**A B C D**	No.	Process
P0	0 0 1 2	0 0 1 2	0 0 0 0	1 5 2 0	1	P_0
P1	1 0 0 0	1 7 5 0	0 7 5 0	1 5 3 2	2	P_2
P2	1 3 5 4	2 3 5 6	1 0 0 2	2 8 8 6	3	P_3
P3	0 6 3 2	0 6 5 2	0 0 2 0	2 14 11 8	4	P_4
P4	0 0 1 4	0 6 5 6	0 6 4 2	2 14 12 12	5	P_1
Number of available instances after all processes get resource allocation →				3 14 12 12	$P_0 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_1$	

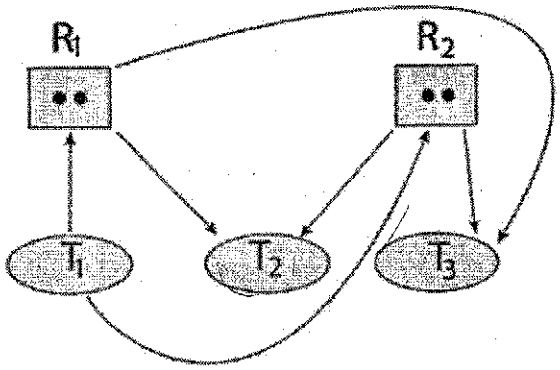
b) Is the system in a safe state? Explain **(5 points)**

Yes, the system is in a safe state.

Because there is a sequence exist to finish all processes.

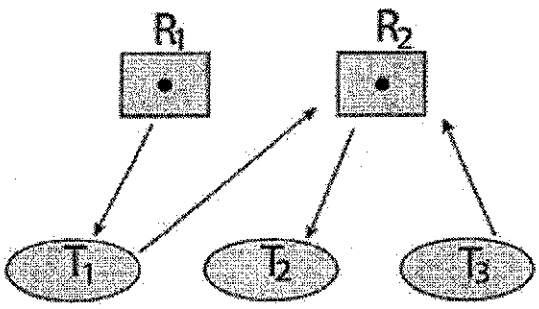
7. Describe which resource allocation graph depict deadlock. For those situations which are deadlocked, provide cycle of threads and resources. If there is no deadlock illustrate the order in which the threads may complete execution.

a) **5 points**



$T_2 \rightarrow T_1 \rightarrow T_3$

b) **5 points**



There is only one instance per resource type, so there may be deadlock.

T_1 holds an instance of R_1
T_2 holds an instance of R_2
T_2 request an instance of R_2
But there is no cycle.
$T_2 \rightarrow T_3 \rightarrow T_1$

$T_1 \rightarrow T_2$
$\uparrow$
T_3

The last page of Exam 2