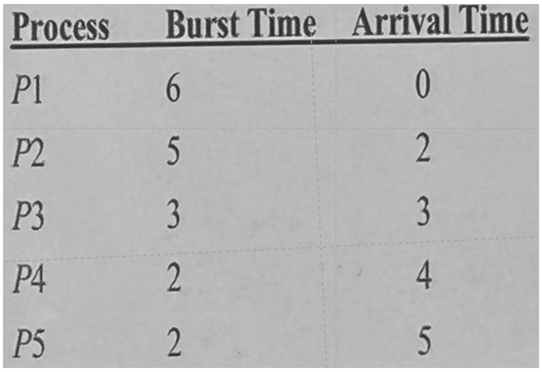
1. Draw the parent-child relationship and explain.

```
main() {
    r = fork();
    if(r<0) exec("programA");
    else if(r>0){
        fork(); exec("programC");
        fork(); exec("programA");
    }
    else{
        if(fork()==0){exec("programB"); fork();}
        else{fork();exec("programC"); fork();}
    }
    while(1);
}
```

2. What not to do when updating shared memory
   (a) access local var
   (b) access global var
   (c) access heap
   (d) open file
   (e) read file

3. Please describe RPC, its relationship with message passing, and the role of stubs in RPC.

4. Given:



| Process | Burst Time | Arrival Time |
|---|---|---|
| *P1* | 6 | 0 |
| *P2* | 5 | 2 |
| *P3* | 3 | 3 |
| *P4* | 2 | 4 |
| *P5* | 2 | 5 |

(a) Draw the Gantt chart illustrating the execution of these processes using non-preemptive SJF, and list the waiting time of each process.
(b) using preemptive SJF, draw the Gantt Chart and list Throughput times of each process

5. For operating system structures, please describe why the modular based approach is more practical than the layered approach.

6. About processor affinity

(a) What is processor affinity?

(b) Why load balancing often counteracts the benefits of processor affinity?

7. About the microkernel

(a) Please briefly describe the microkernel system structure.

(b) Please describe two benefits and one drawback of microkernel.

8. About NUMA

(a) Please NUMA and its advantage(s).

(b) Please explain why does the complexity of OS design increase with NUMA.

9. Which of the following situation(s) always cause a transition from the user mode to the kernel mode?

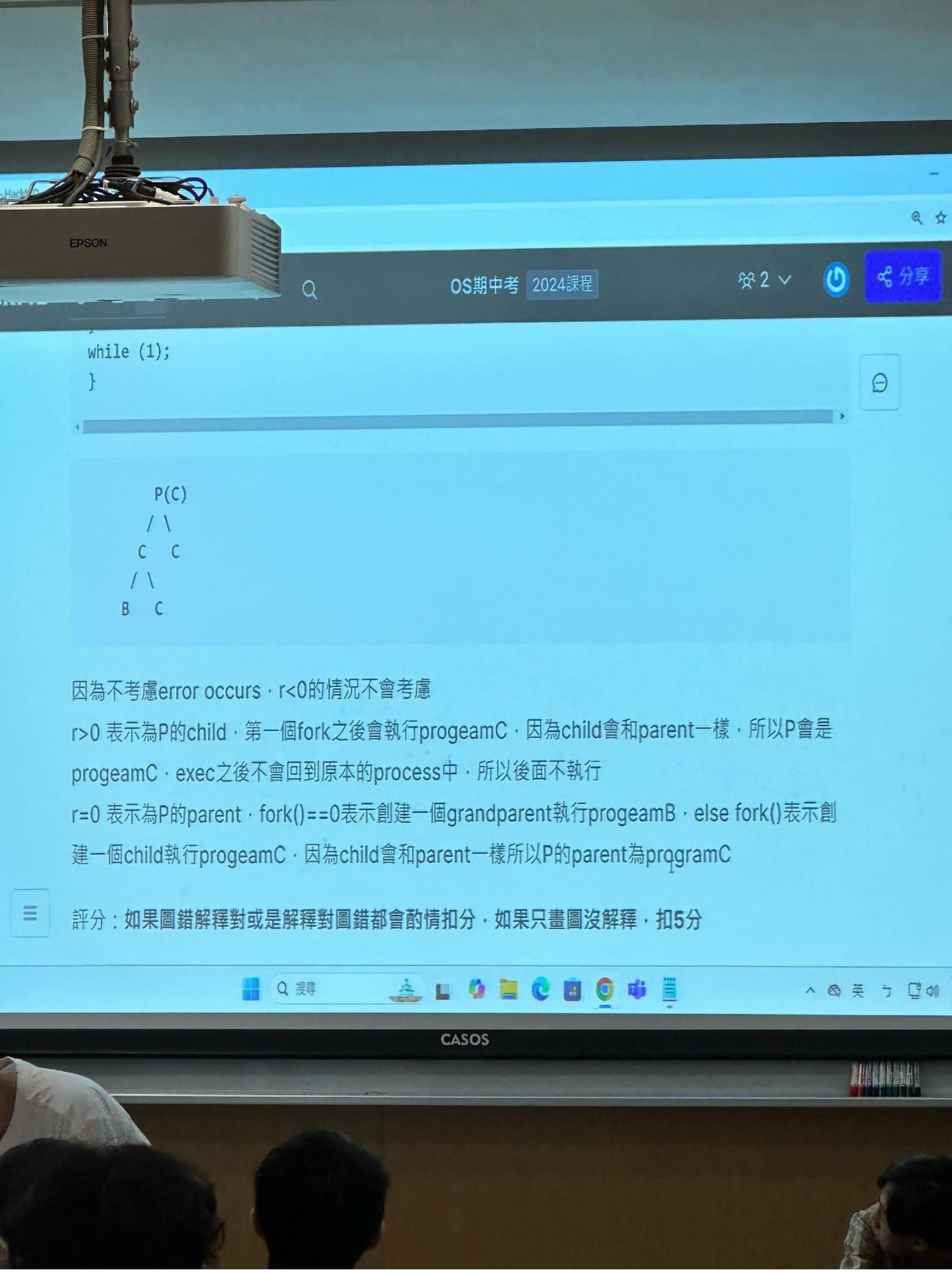
(a) a timer interrupt occurs but the time quantum of the process has not expired

(b) a timer interrupt occurs and the time quantum of the process has expired

(c) a process issues a system call

(d) the process opens a file

(e) the process reads a file



```
while (1);
}
```

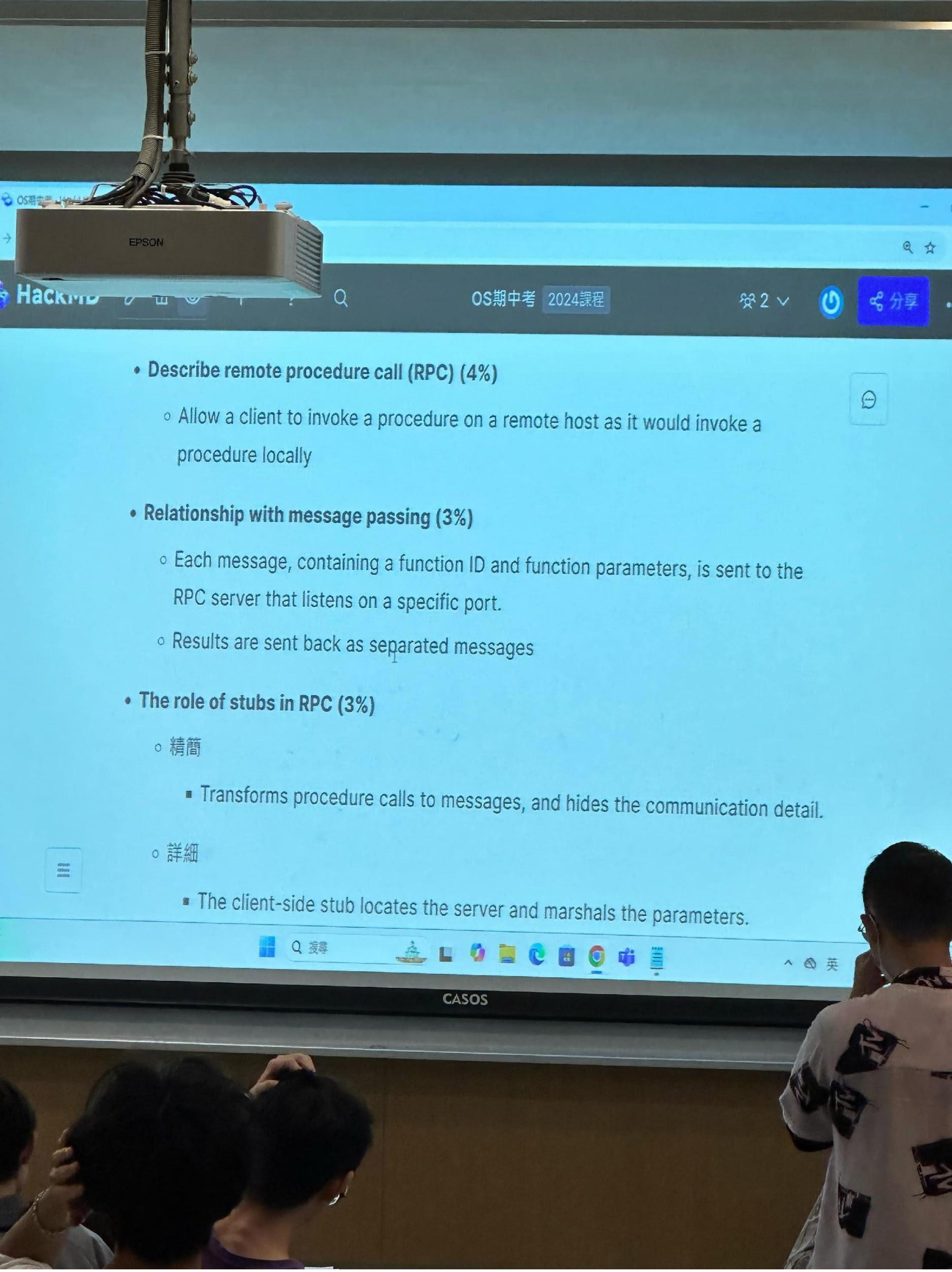
```
      P(C)
      / \
     C   C
    / \
   B   C
```

因為不考慮error occurs，r<0的情況不會考慮

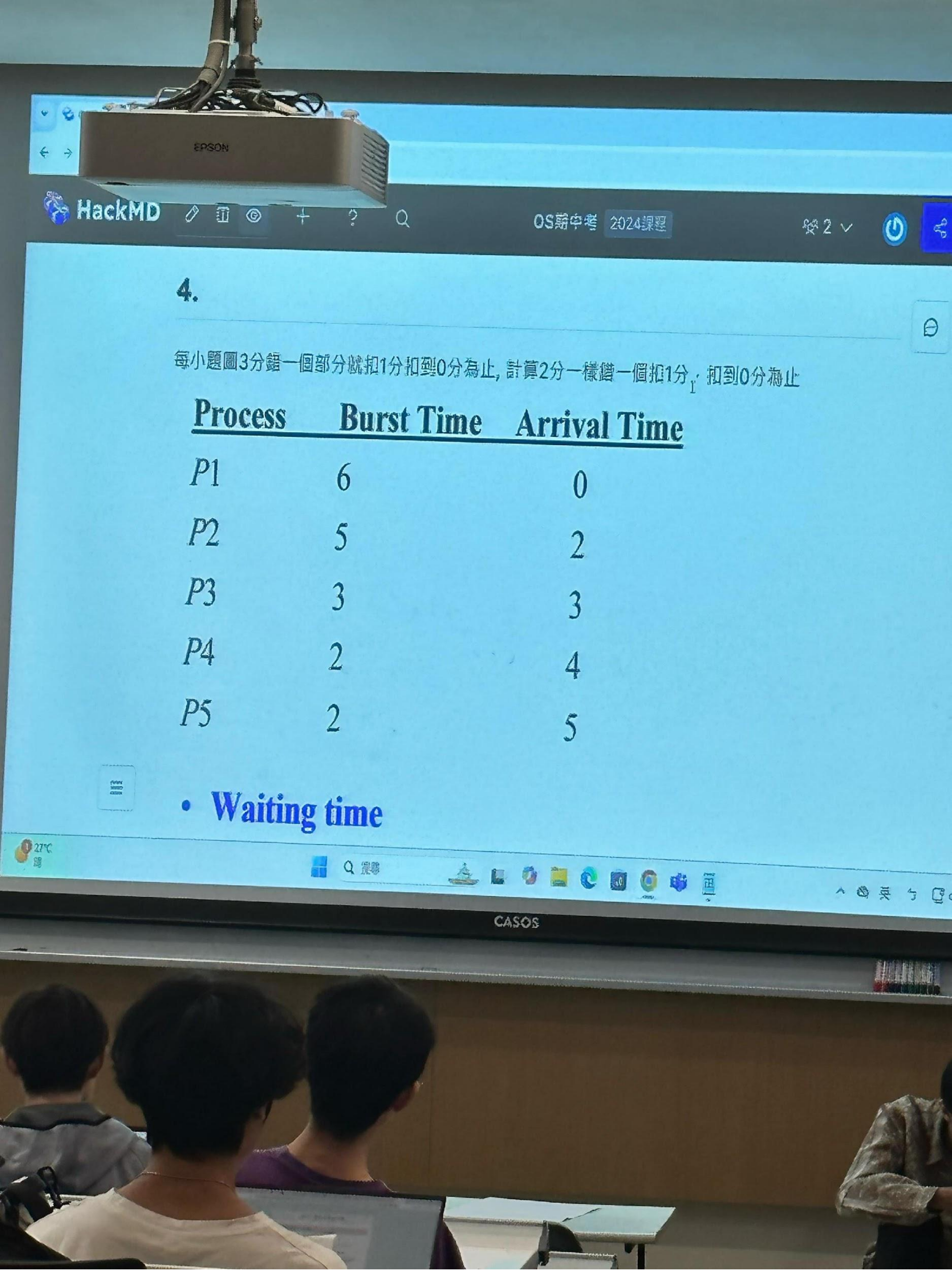
r>0 表示為P的child，第一個fork之後會執行progeamC，因為child會和parent一樣，所以P會是progeamC，exec之後不會回到原本的process中，所以後面不執行

r=0 表示為P的parent，fork()==0表示創建一個grandparent執行progeamB，else fork()表示創建一個child執行progeamC，因為child會和parent一樣所以P的parent為programC

評分：**如果圖錯解釋對或是解釋對圖錯都會酌情扣分，如果只畫圖沒解釋，扣5分**



- **Describe remote procedure call (RPC) (4%)**
  - Allow a client to invoke a procedure on a remote host as it would invoke a procedure locally
- **Relationship with message passing (3%)**
  - Each message, containing a function ID and function parameters, is sent to the RPC server that listens on a specific port.
  - Results are sent back as separated messages
- **The role of stubs in RPC (3%)**
  - 精簡
    - Transforms procedure calls to messages, and hides the communication detail.
  - 詳細
    - The client-side stub locates the server and marshals the parameters.

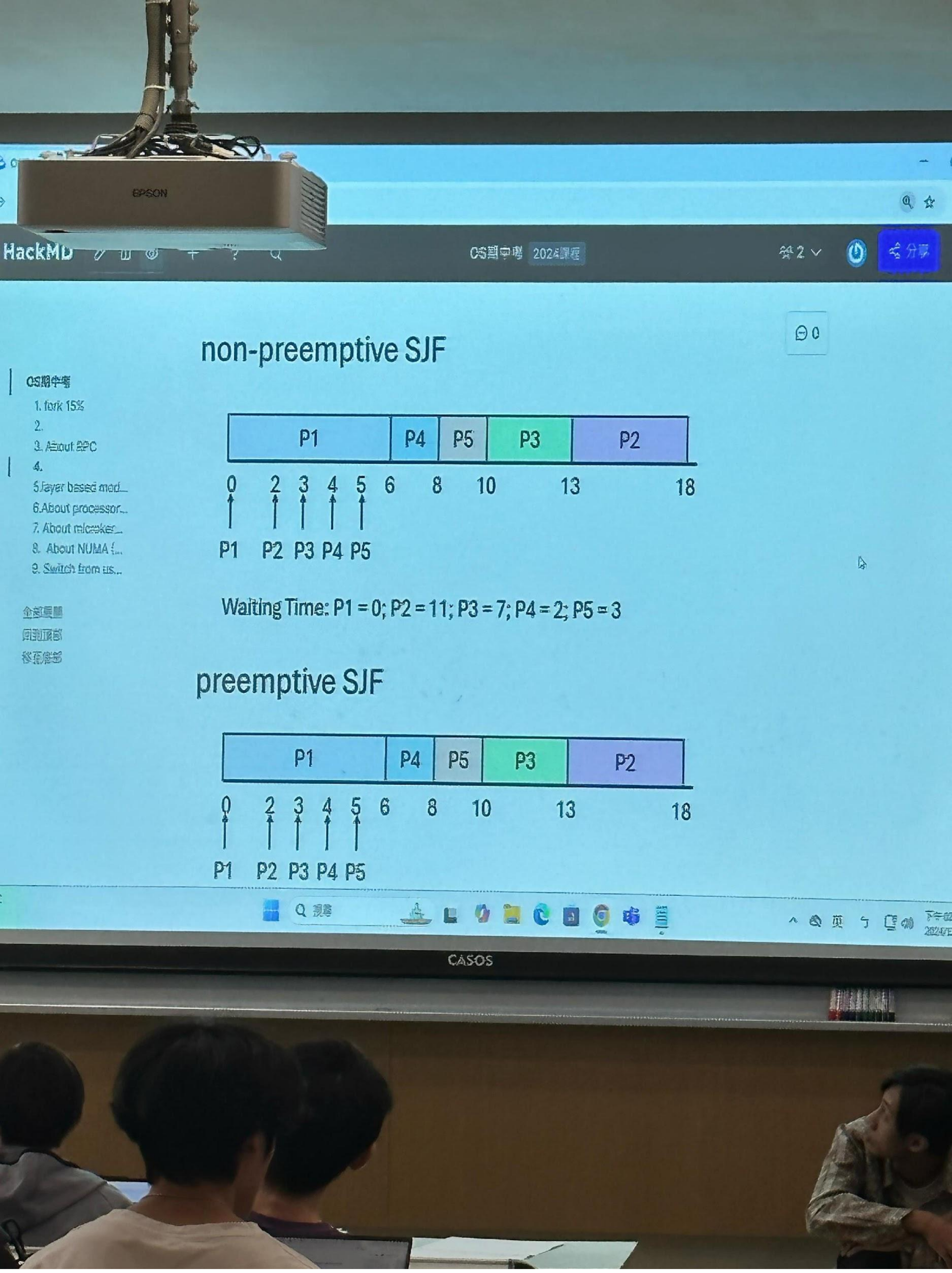


## 4.

每小題圖3分錯一個部分就扣1分扣到0分為止, 計算2分一樣錯一個扣1分，扣到0分為止

| Process | Burst Time | Arrival Time |
|---|---|---|
| *P*1 | 6 | 0 |
| *P*2 | 5 | 2 |
| *P*3 | 3 | 3 |
| *P*4 | 2 | 4 |
| *P*5 | 2 | 5 |

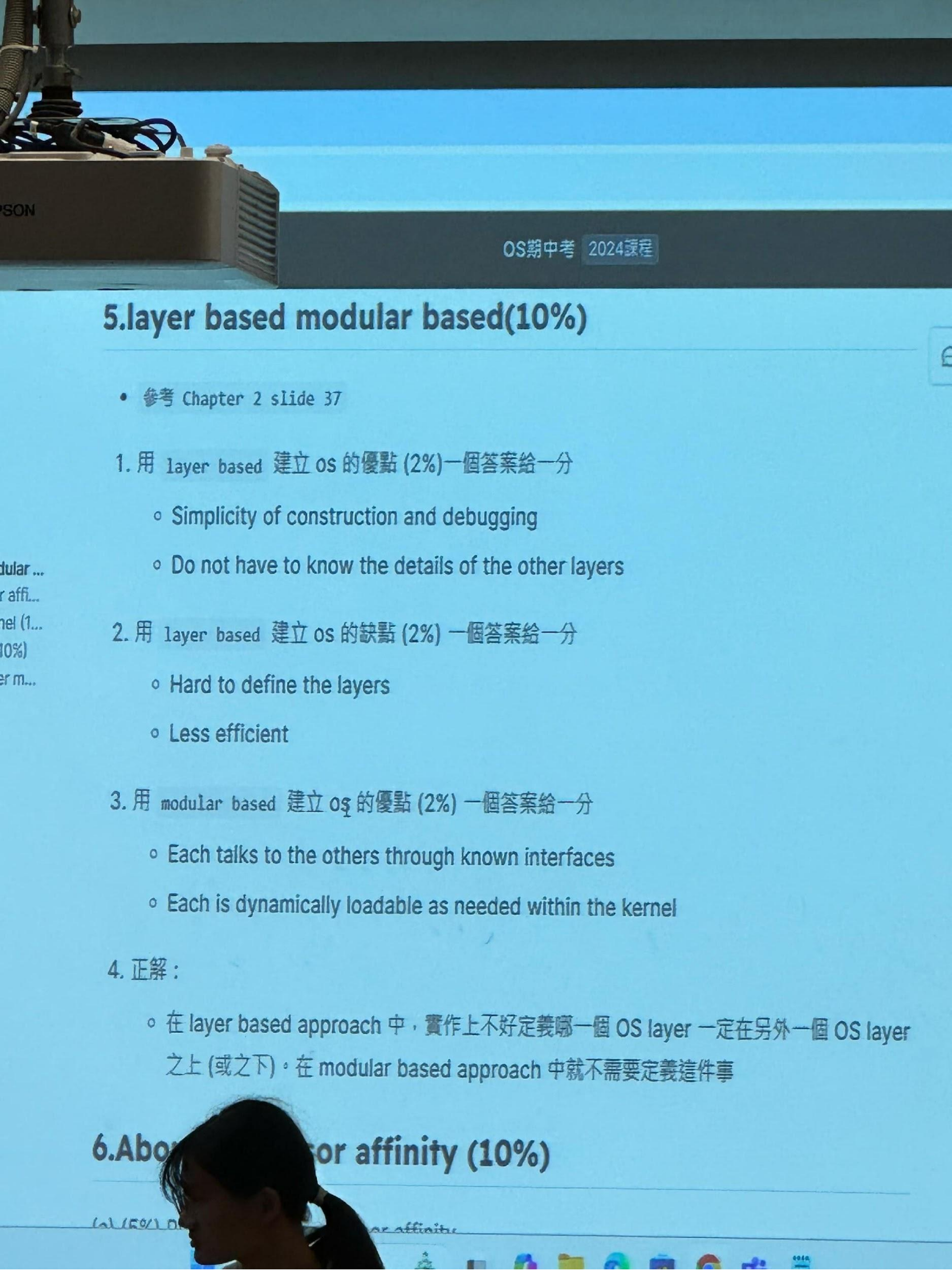
- Waiting time



## non-preemptive SJF

Waiting Time: P1 = 0; P2 = 11; P3 = 7; P4 = 2; P5 = 3

## preemptive SJF



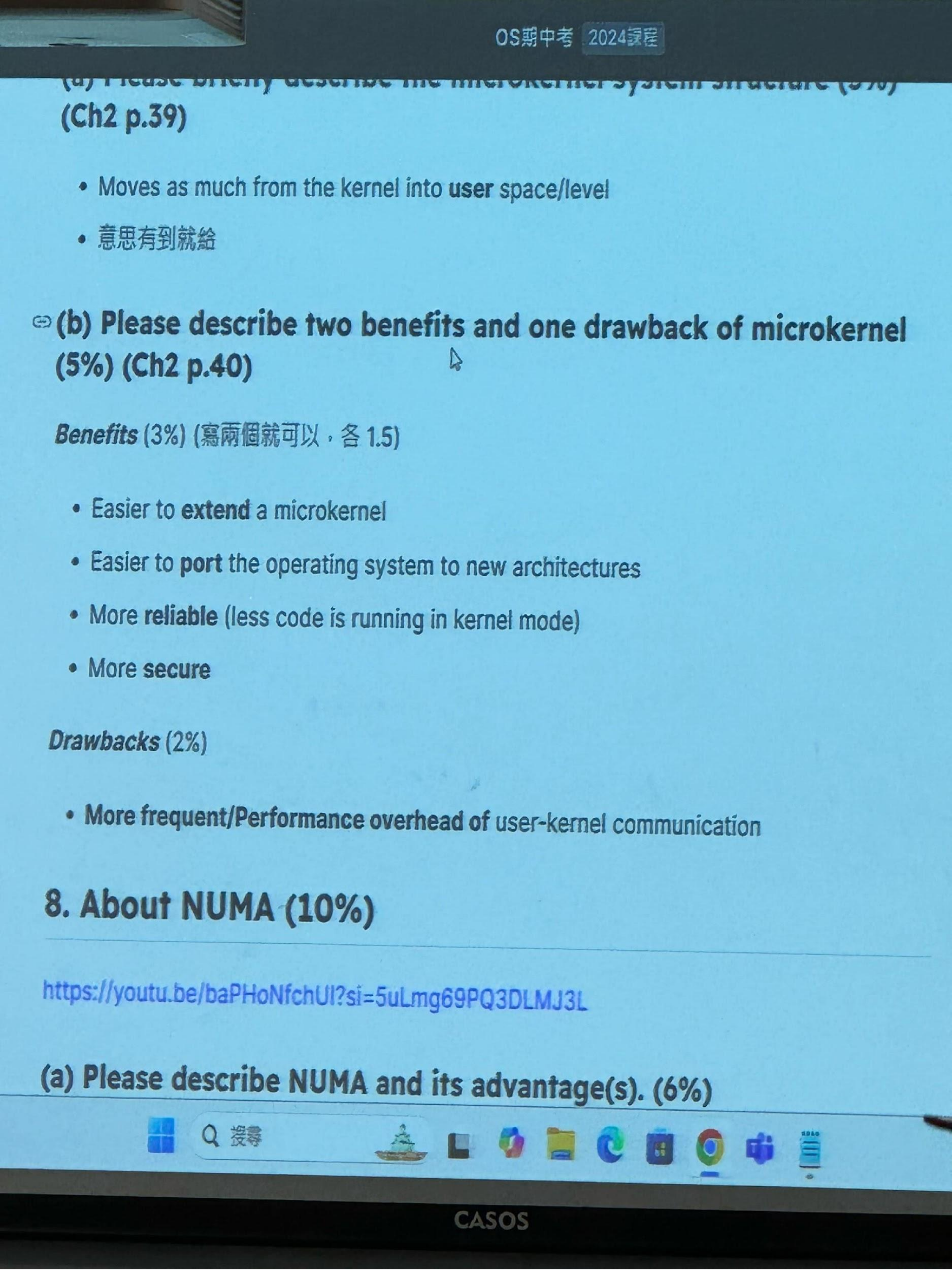
## 5.layer based modular based(10%)

- 參考 Chapter 2 slide 37

1. 用 `layer based` 建立 os 的優點 (2%)一個答案給一分
   - Simplicity of construction and debugging
   - Do not have to know the details of the other layers
2. 用 `layer based` 建立 os 的缺點 (2%) 一個答案給一分
   - Hard to define the layers
   - Less efficient
3. 用 `modular based` 建立 os 的優點 (2%) 一個答案給一分
   - Each talks to the others through known interfaces
   - Each is dynamically loadable as needed within the kernel
4. 正解：
   - 在 layer based approach 中，實作上不好定義哪一個 OS layer 一定在另外一個 OS layer 之上 (或之下)。在 modular based approach 中就不需要定義這件事

## 6.Abo sor affinity (10%)

(a) (5%) D or affinity



### (a) Please briefly describe the microkernel system structure (5%) (Ch2 p.39)

- Moves as much from the kernel into **user** space/level
- 意思有到就給

### (b) Please describe two benefits and one drawback of microkernel (5%) (Ch2 p.40)

***Benefits*** (3%) (寫兩個就可以，各 1.5)

- Easier to **extend** a microkernel
- Easier to **port** the operating system to new architectures
- More **reliable** (less code is running in kernel mode)
- More **secure**

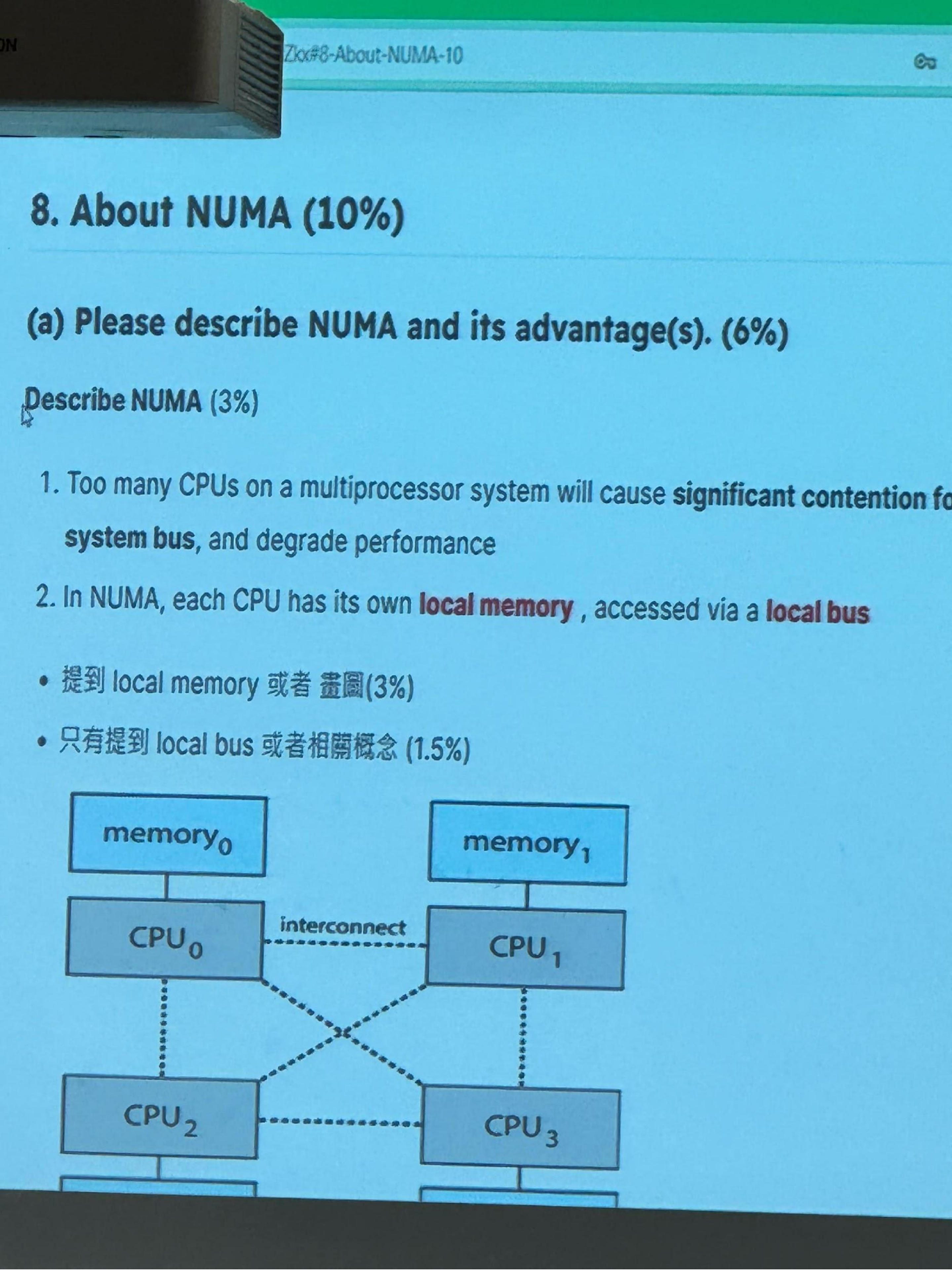
***Drawbacks*** (2%)

- **More frequent/Performance overhead of** user-kernel communication

## 8. About NUMA (10%)

https://youtu.be/baPHoNfchUI?si=5uLmg69PQ3DLMJ3L

### (a) Please describe NUMA and its advantage(s). (6%)



ON

Zkx#8-About-NUMA-10

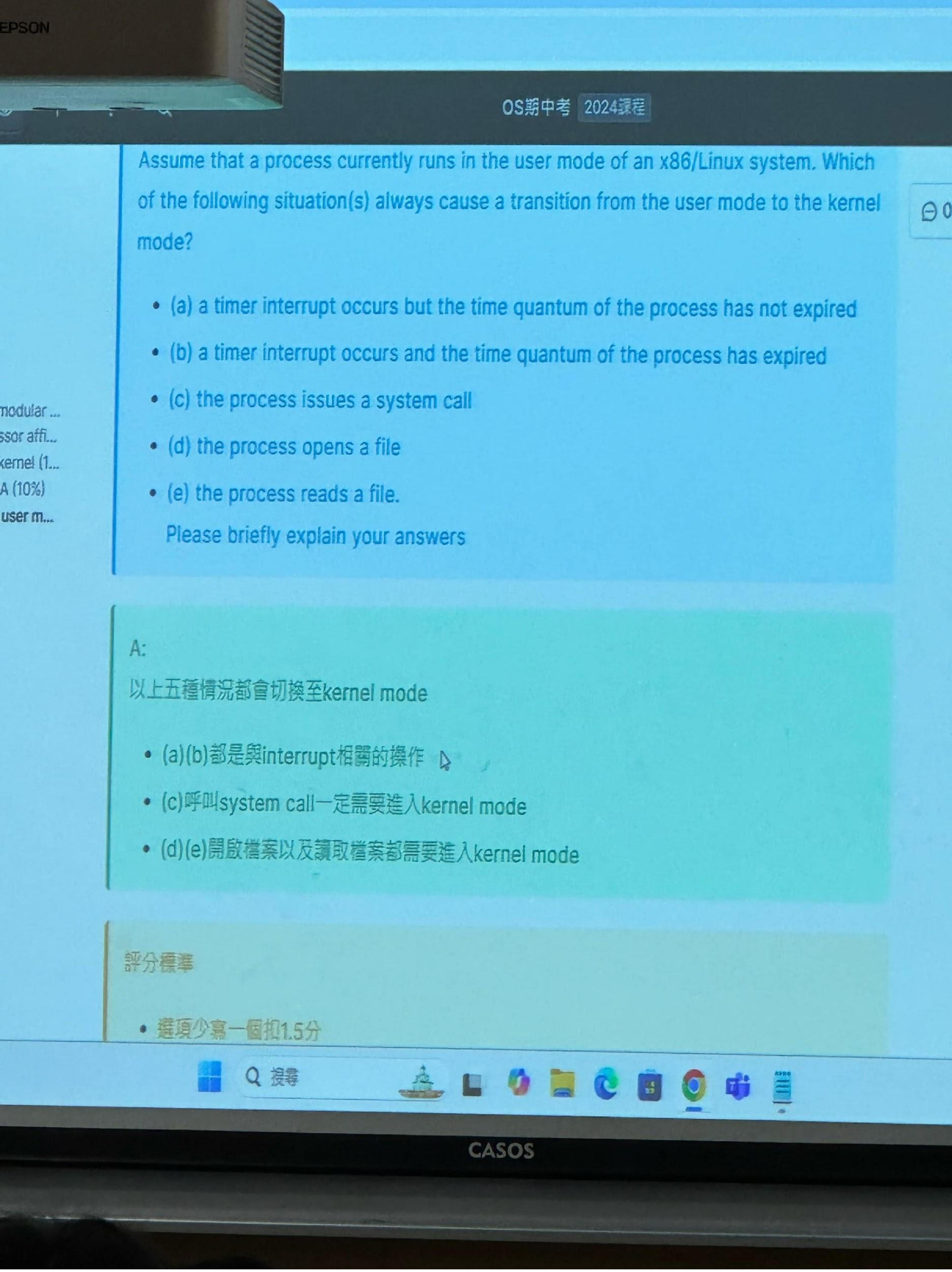
## 8. About NUMA (10%)

### (a) Please describe NUMA and its advantage(s). (6%)

Describe NUMA (3%)

1. Too many CPUs on a multiprocessor system will cause **significant contention fo** **system bus**, and degrade performance
2. In NUMA, each CPU has its own **local memory** , accessed via a **local bus**

- 提到 local memory 或者 畫圖(3%)
- 只有提到 local bus 或者相關概念 (1.5%)



Assume that a process currently runs in the user mode of an x86/Linux system. Which of the following situation(s) always cause a transition from the user mode to the kernel mode?

- (a) a timer interrupt occurs but the time quantum of the process has not expired
- (b) a timer interrupt occurs and the time quantum of the process has expired
- (c) the process issues a system call
- (d) the process opens a file
- (e) the process reads a file.

Please briefly explain your answers

A:

以上五種情況都會切換至kernel mode

- (a)(b)都是與interrupt相關的操作
- (c)呼叫system call一定需要進入kernel mode
- (d)(e)開啟檔案以及讀取檔案都需要進入kernel mode

評分標準

- 選項少寫一個扣1.5分