**Project3**

**[ PART I ] NPCS Implementation**

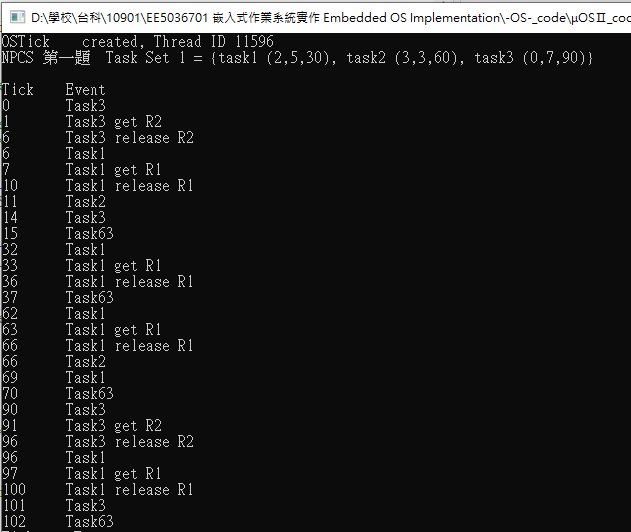
M10907324 吳俊逸

**1.The screenshot result (with the given format) of the two task sets. (Time tick 0-**

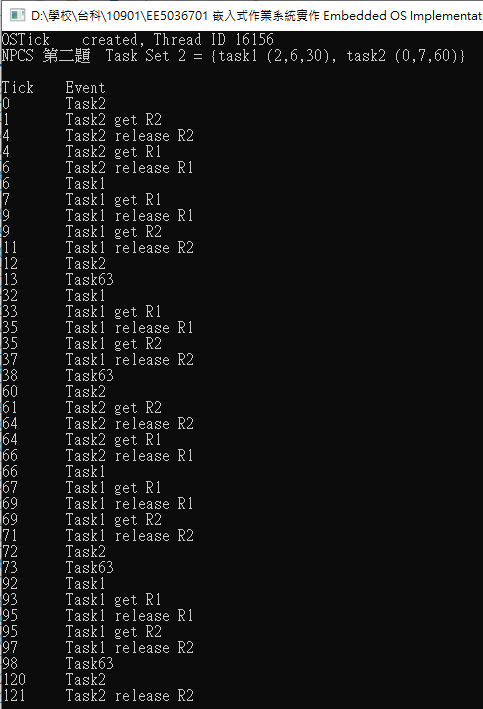
**100)**

The NPCS of Output Result:

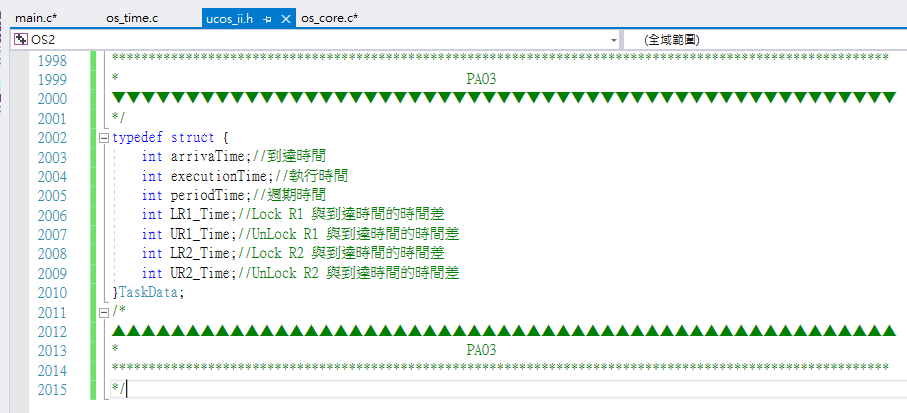
Task Set 1 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}

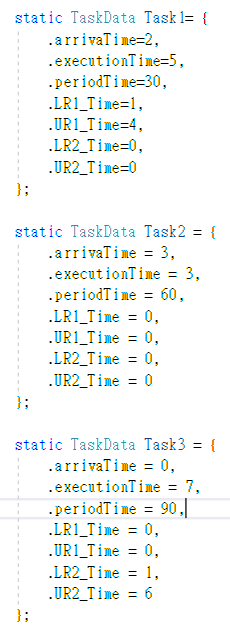
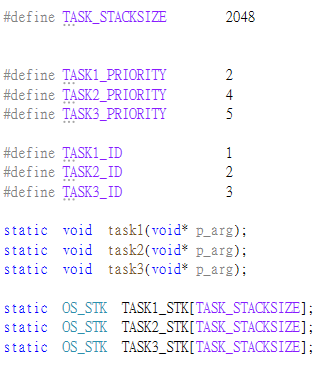


Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}

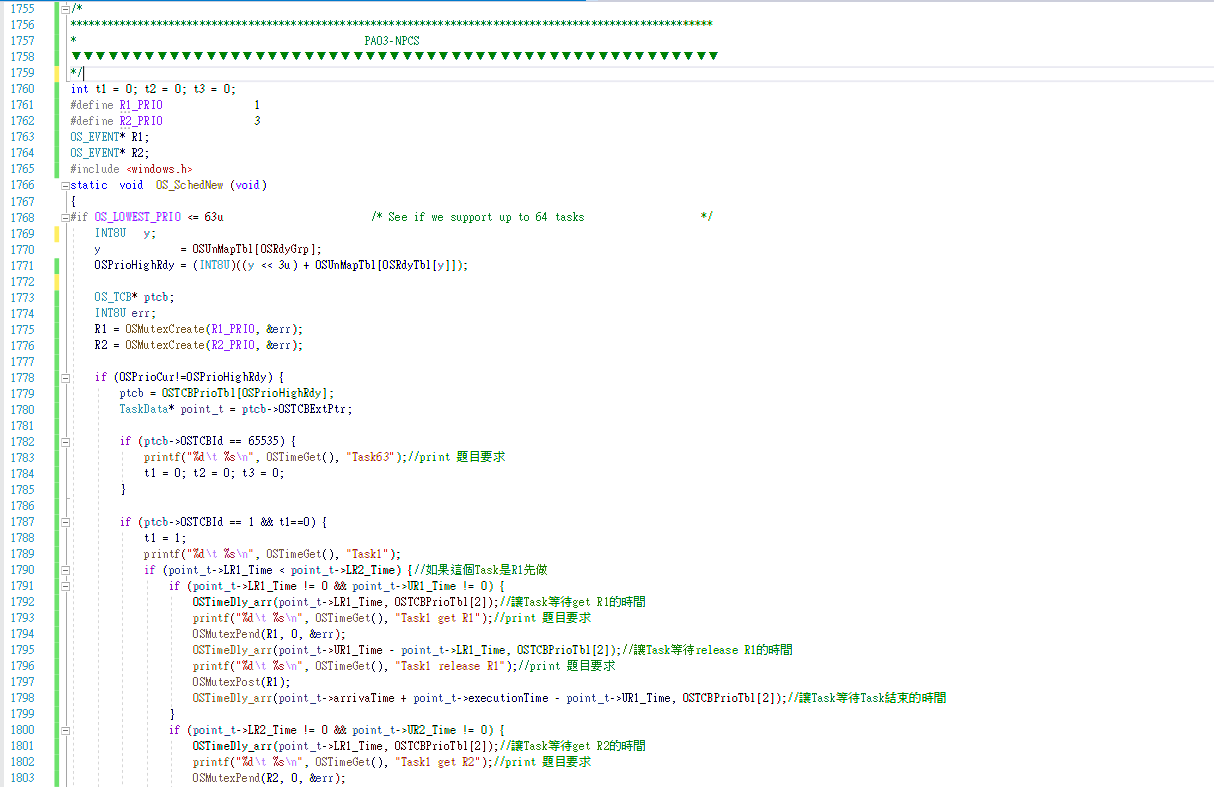


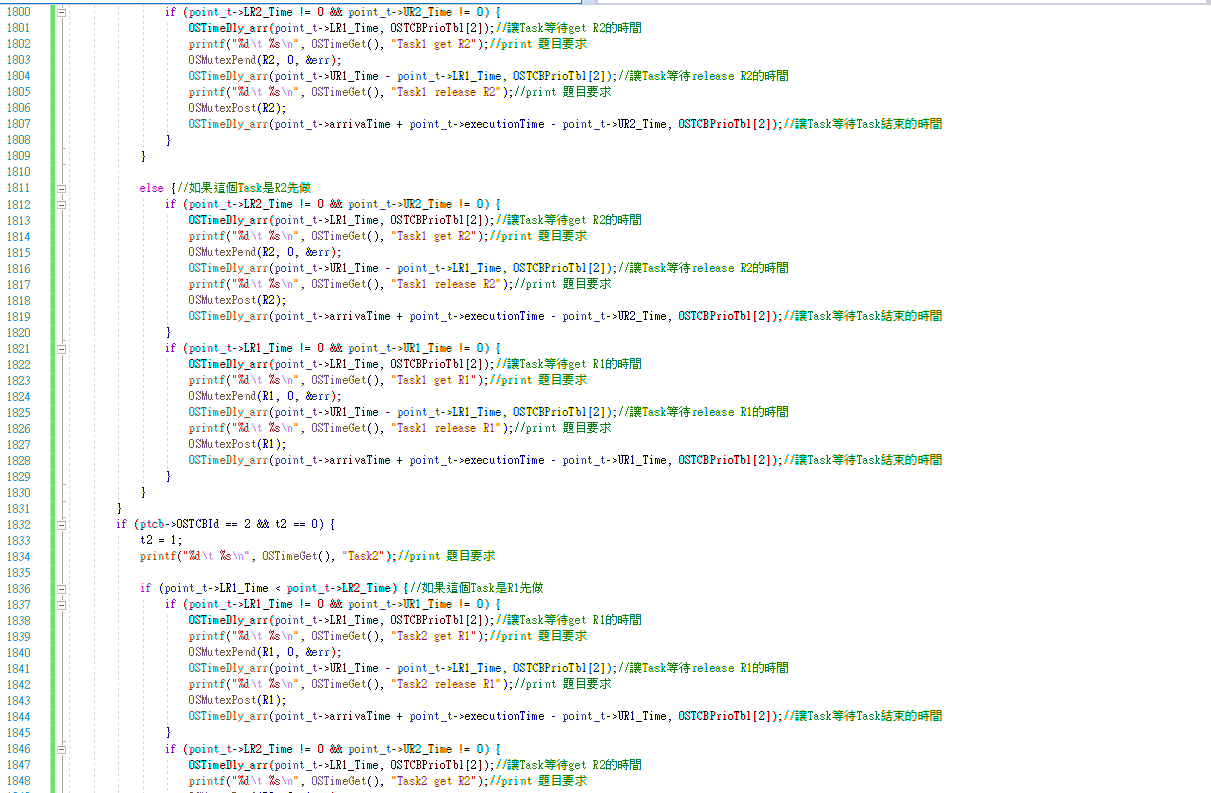
2.A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please ATTACH the screenshot of the code and MARK the modified part).

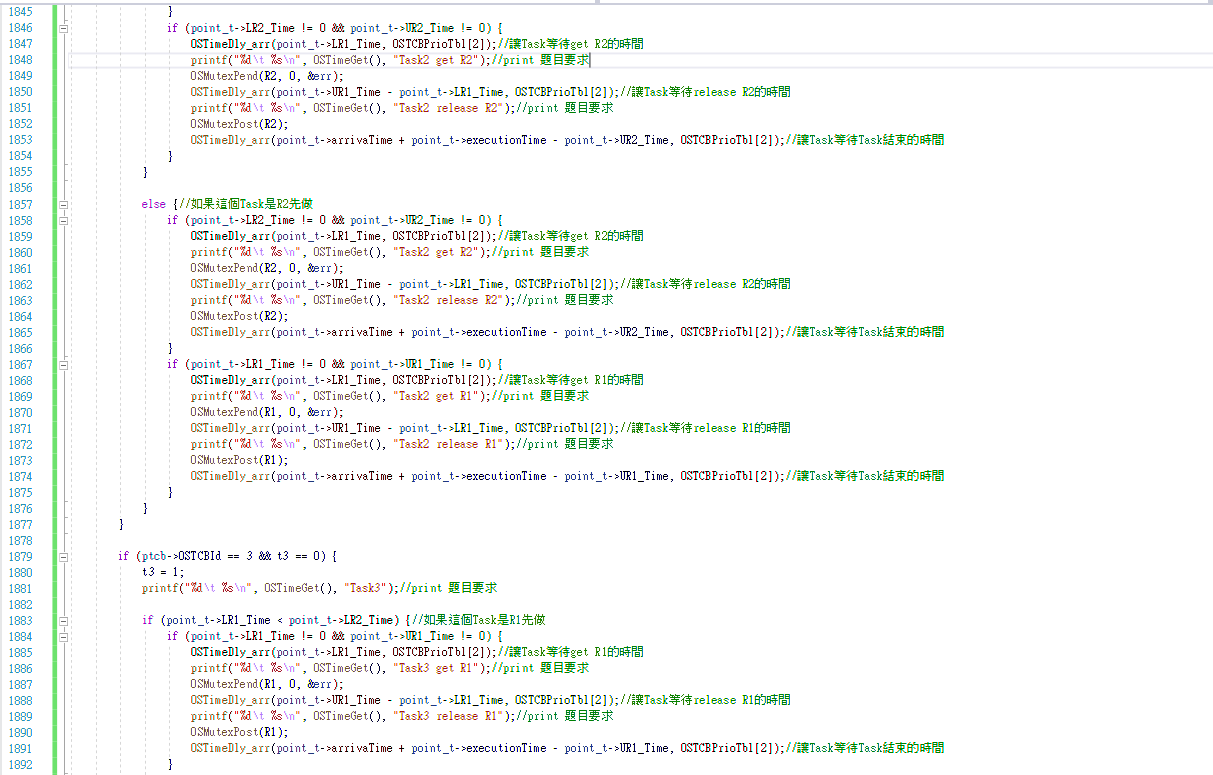


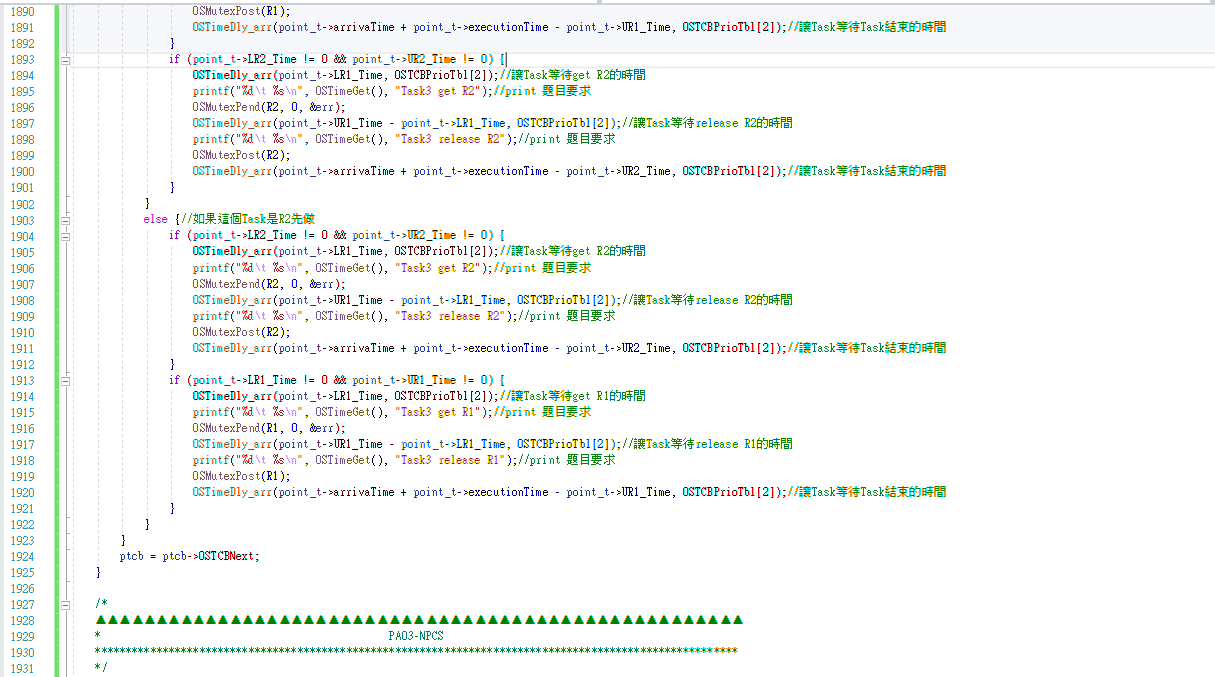


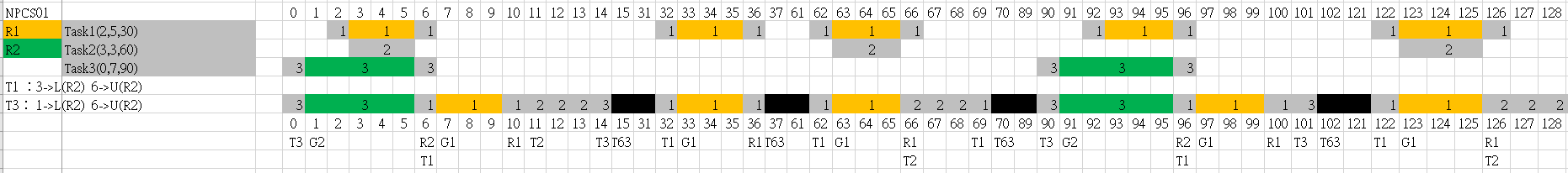


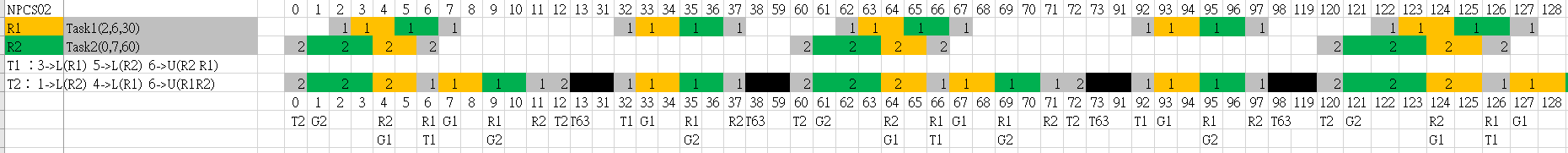








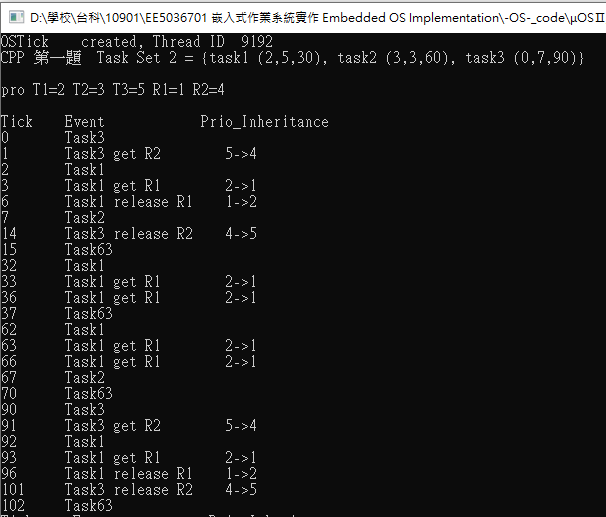




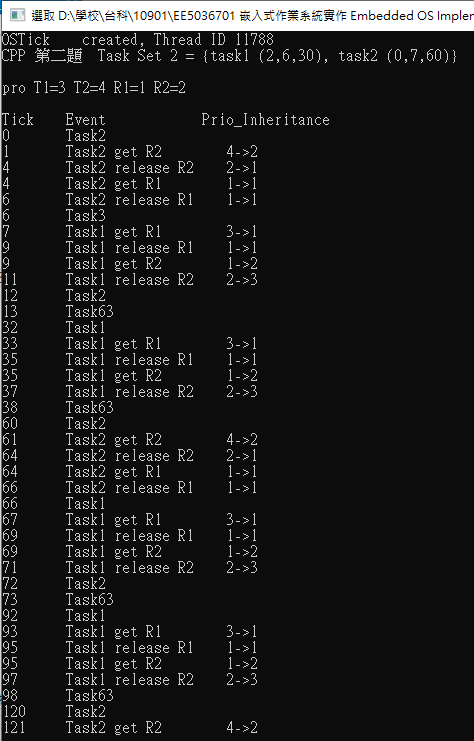
**[ PART II ] CPP Implementation**

1. **The screenshot result (with the given format) of the two task sets. (Time tick 0- 100)**

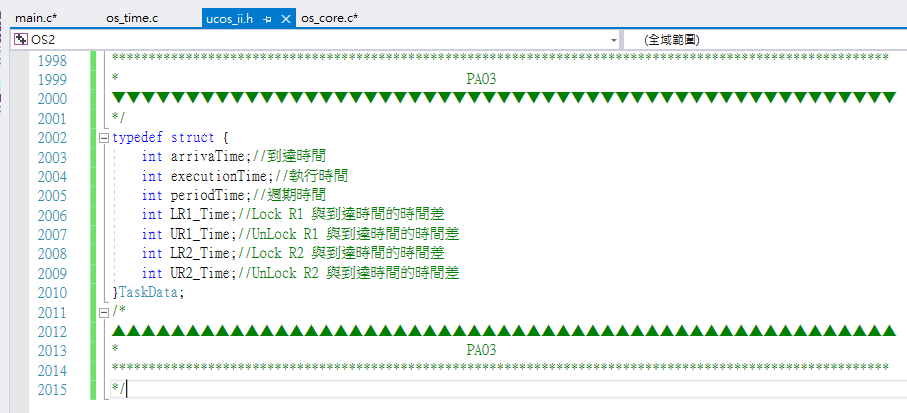
Task Set 1 = {task1 (2,5,30), task2 (3,3,60), task3 (0,7,90)}

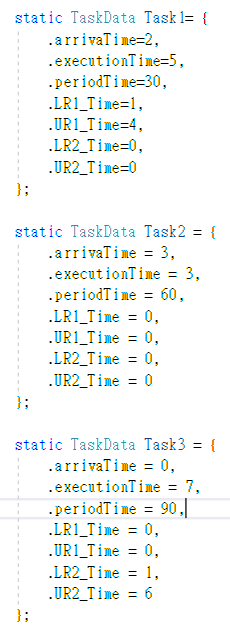
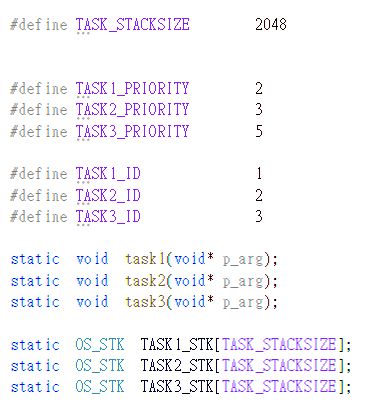
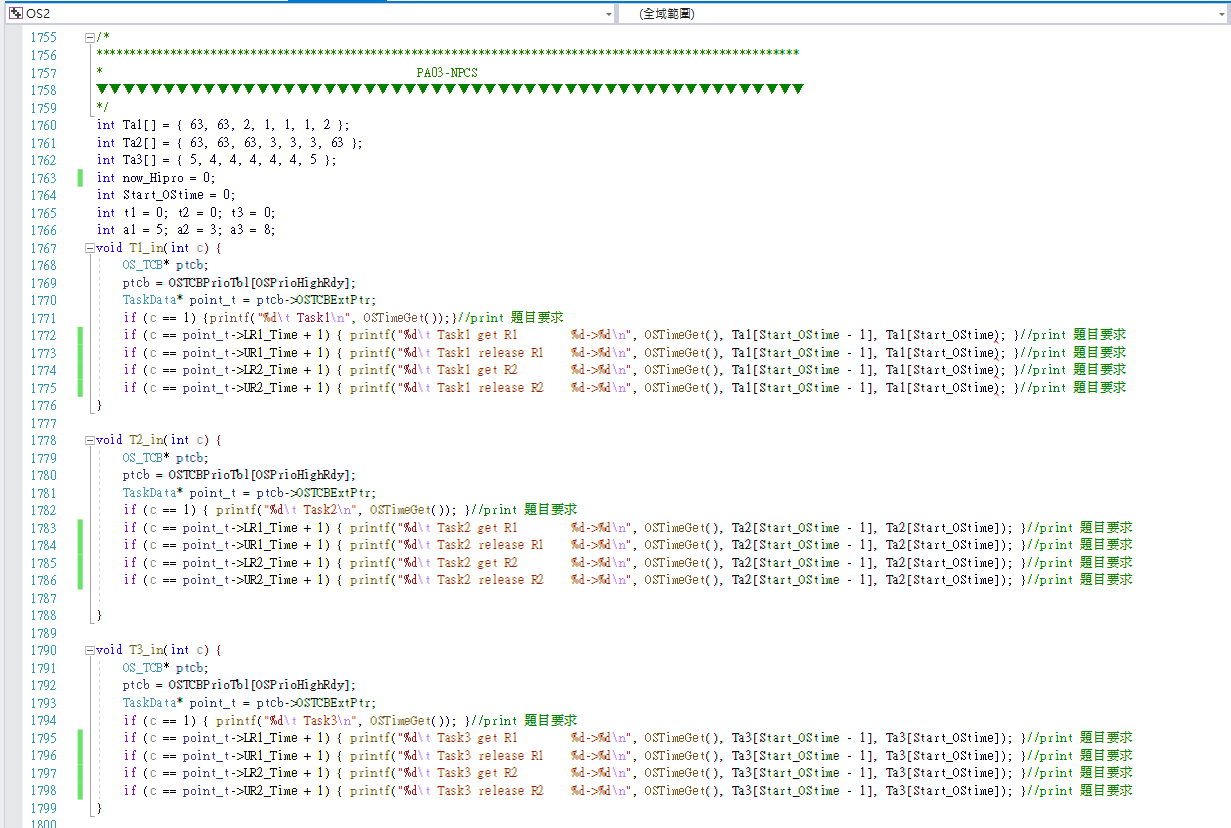
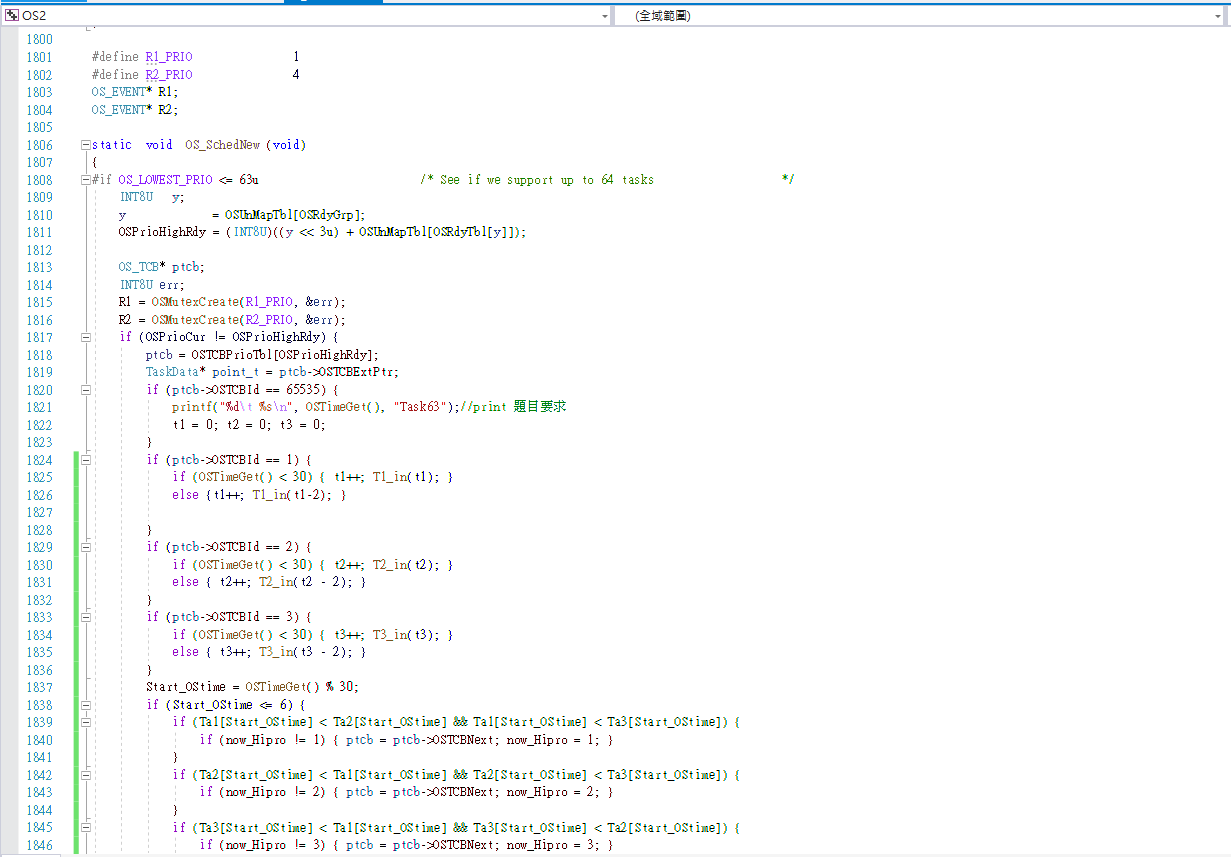
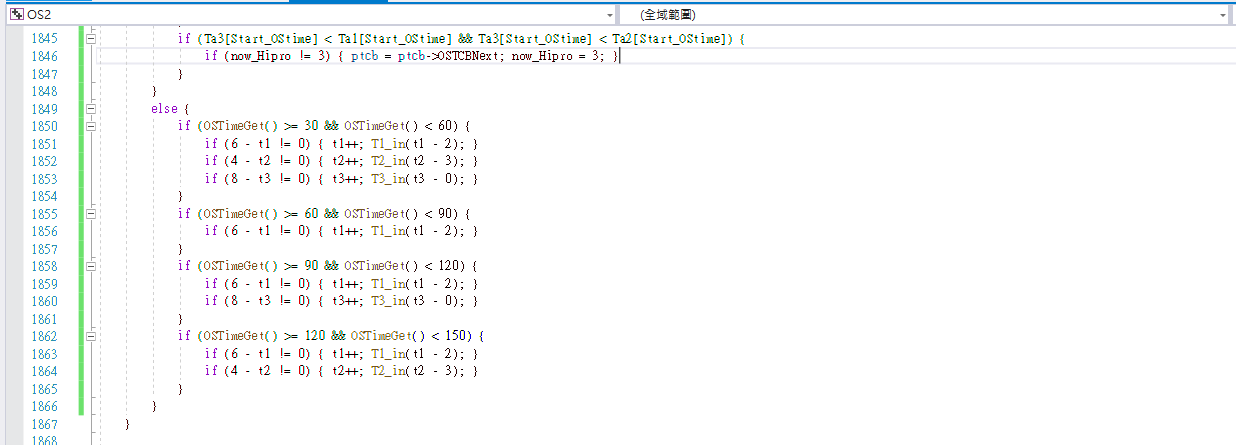


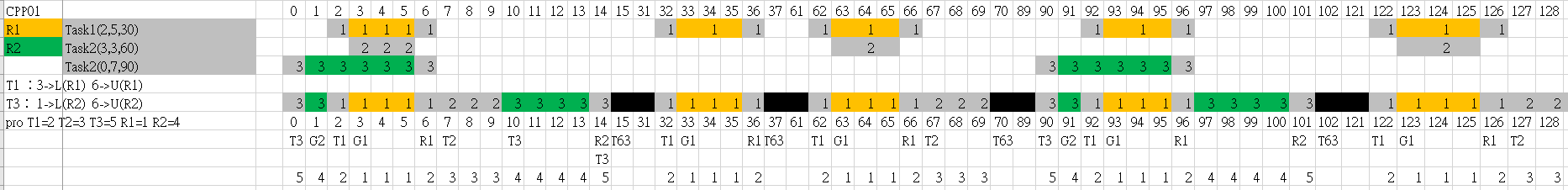
Task Set 2 = {task1 (2,6,30), task2 (0,7,60)}

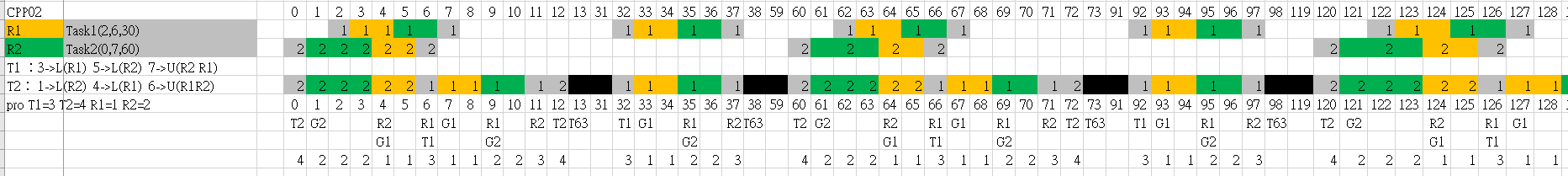


**2.A report that describes your implementation, including scheduling results of two task sets, modified functions, data structure, etc. (please ATTACH the screenshot of the code and MARK the modified part).**







**[ PART III ] Performance Analysis**

**1.Compare the scheduling behaviors between NPCS and CPP with the results of PART I and PART II. (5%)**

**NPCS：**

If a job is using a resource, it won’t be preempted by any other jobs

– Even if there are no resource conflicts

– Effectively the job runs at the highest priority

**CPP：**

If R is in use, T is blocked.

If R is free, R is allocated to T. T's execution priority is raised to the priority ceiling of R if that is higher. At any given time, T's execution priority equals the highest priority ceiling of all its held resources.

**2.Explain how NPCS and CPP avoid the deadlock problem. (5%)**

**NPCS：**

Deadlocks never occur because any job holding resources can not be preempted

**CPP：**

T's priority is assigned the next-highest priority ceiling of another resource when the resource with the highest priority ceiling is released. The task returns to its assigned priority after it has released all resources.