Project3

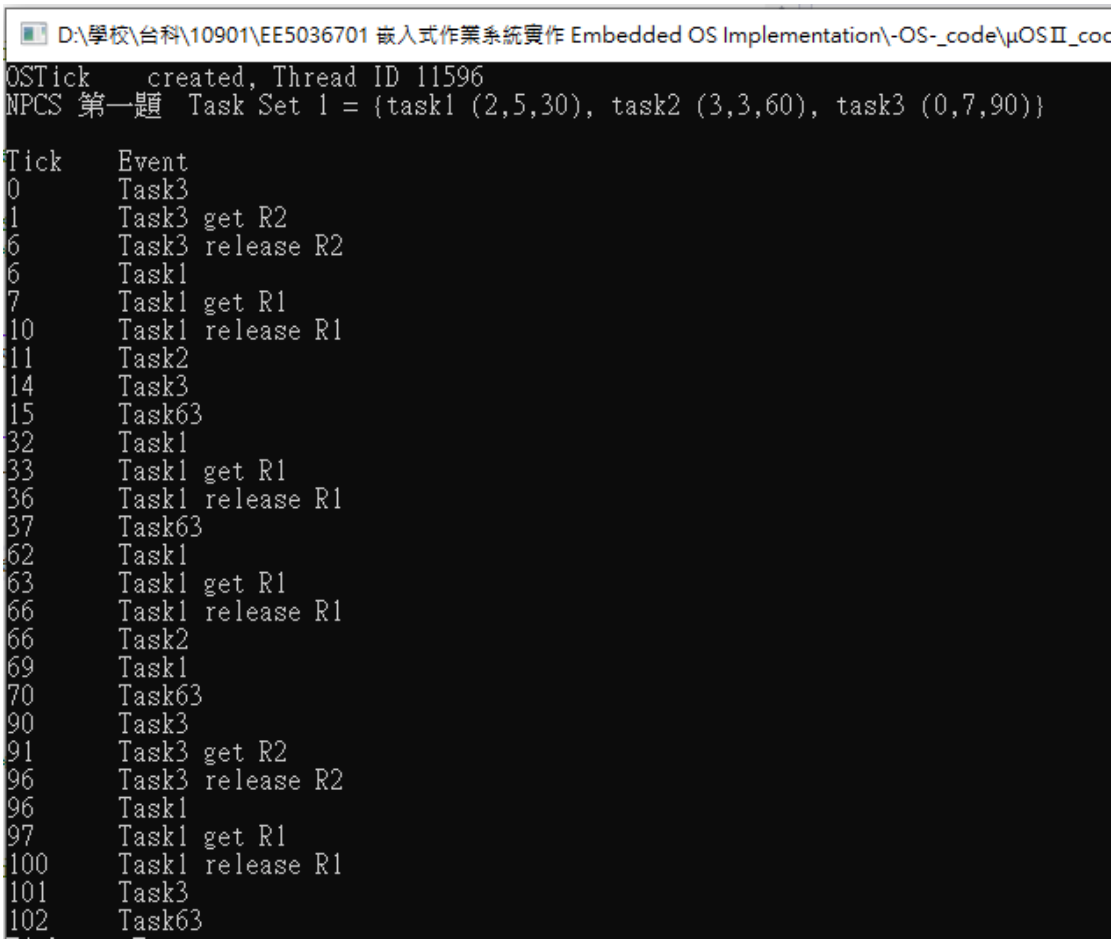
[PART I] NPCS Implementation

M10907324 吳俊逸

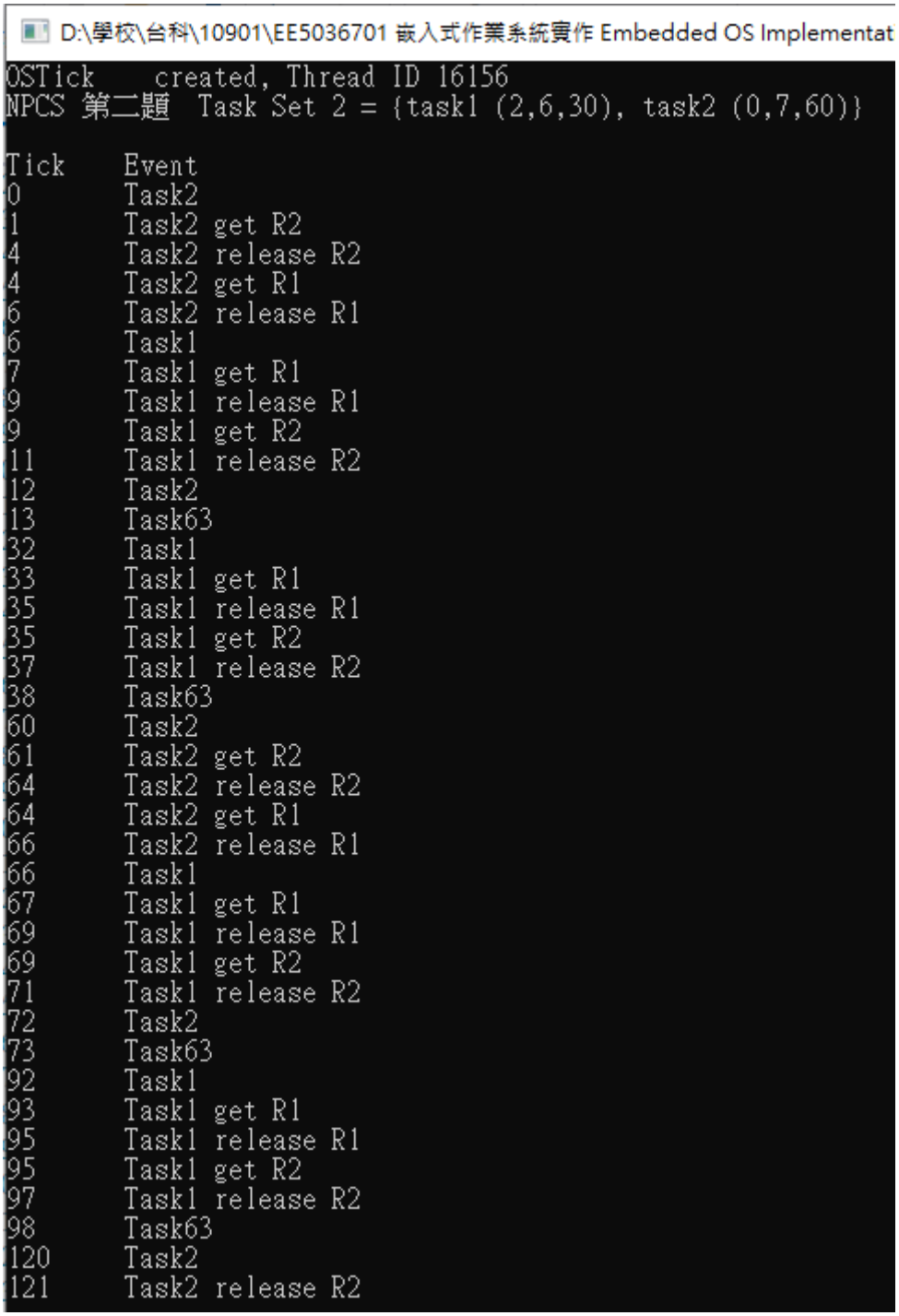
1.The screenshot result (with the given format) of the two task sets. (Time tick 0-100)

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).

main.c* os_time.c ucos_ii.h os_core.c*

OS2 (全域範圍)

```
1998  *********************************************************************************************************
1999  *                                                PA03
2000  ▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼
2001  */
2002  typedef struct {
2003      int arrivaTime;//到達時間
2004      int executionTime;//執行時間
2005      int periodTime;//週期時間
2006      int LR1_Time;//Lock R1 與到達時間的時間差
2007      int UR1_Time;//UnLock R1 與到達時間的時間差
2008      int LR2_Time;//Lock R2 與到達時間的時間差
2009      int UR2_Time;//UnLock R2 與到達時間的時間差
2010  }TaskData;
2011  /*
2012  ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲
2013  *                                                PA03
2014  *********************************************************************************************************
2015  */
```

```
static TaskData Task1= {
    .arrivaTime=2,
    .executionTime=5,
    .periodTime=30,
    .LR1_Time=1,
    .UR1_Time=4,
    .LR2_Time=0,
    .UR2_Time=0
};

static TaskData Task2 = {
    .arrivaTime = 3,
    .executionTime = 3,
    .periodTime = 60,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 0,
    .UR2_Time = 0
};

static TaskData Task3 = {
    .arrivaTime = 0,
    .executionTime = 7,
    .periodTime = 90,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 1,
    .UR2_Time = 6
};
```

```
#define TASK_STACKSIZE        2048


#define TASK1_PRIORITY        2
#define TASK2_PRIORITY        4
#define TASK3_PRIORITY        5

#define TASK1_ID              1
#define TASK2_ID              2
#define TASK3_ID              3

static  void  task1(void* p_arg);
static  void  task2(void* p_arg);
static  void  task3(void* p_arg);

static  OS_STK  TASK1_STK[TASK_STACKSIZE];
static  OS_STK  TASK2_STK[TASK_STACKSIZE];
static  OS_STK  TASK3_STK[TASK_STACKSIZE];
```

```c
*                                                    PA03
▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼
*/
void mywait(int tick)
{
#if OS_CRITICAL_METHOD==3
    OS_CPU_SR cpu_sr = 0;
#endif
    int now, exit;
    OS_ENTER_CRITICAL();
    now = OSTimeGet();
    exit = now + tick;
    OS_EXIT_CRITICAL();
    while (1) {
        if (exit <= OSTimeGet())
            break;
    }
}
void  OSTimeDly_arr(INT32U ticks, OS_TCB* task_123) //把arrival time tick 與 taskName 送進來
{
    INT8U      y;
#if OS_CRITICAL_METHOD == 3u                     /* Allocate storage for CPU status register           */
    OS_CPU_SR  cpu_sr = 0u;
#endif
    if (OSIntNesting > 0u) {                     /* See if trying to call from an ISR                  */
        return;
    }
    if (OSLockNesting > 0u) {                    /* See if called with scheduler locked                */
        return;
    }
    if (ticks > 0u) {                            /* 0 means no delay!                                  */
        OS_ENTER_CRITICAL();
        y = task_123->OSTCBY;        /* Delay current task                                 */
        OSRdyTbl[y] &= (OS_PRIO)~task_123->OSTCBBitX;
        OS_TRACE_TASK_SUSPENDED(task_123);
        if (OSRdyTbl[y] == 0u) {
            OSRdyGrp &= (OS_PRIO)~task_123->OSTCBBitY;
        }
        task_123->OSTCBDly = ticks;              /* Load ticks in TCB                                  */
        OS_TRACE_TASK_DLY(ticks);
        OS_EXIT_CRITICAL();
        //OS_Sched();                              /* Find next task to run!                             */
    }
}
/*
▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲
*                                                    PA03
****************************************************************************************************
*/
```

```
/*
*********************************************************************************************
*                                                PA03-NPCS
▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼
*/
int t1 = 0; t2 = 0; t3 = 0;
#define R1_PRIO               1
#define R2_PRIO               3
OS_EVENT* R1;
OS_EVENT* R2;
#include <windows.h>
static  void  OS_SchedNew (void)
{
#if OS_LOWEST_PRIO <= 63u                        /* See if we support up to 64 tasks                   */
    INT8U   y;
    y             = OSUnMapTbl[OSRdyGrp];
    OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

    OS_TCB* ptcb;
    INT8U err;
    R1 = OSMutexCreate(R1_PRIO, &err);
    R2 = OSMutexCreate(R2_PRIO, &err);

    if (OSPrioCur!=OSPrioHighRdy) {
        ptcb = OSTCBPrioTbl[OSPrioHighRdy];
        TaskData* point_t = ptcb->OSTCBExtPtr;

        if (ptcb->OSTCBId == 65535) {
            printf("%d\t %s\n", OSTimeGet(), "Task63");//print 題目要求
            t1 = 0; t2 = 0; t3 = 0;
        }

        if (ptcb->OSTCBId == 1 && t1==0) {
            t1 = 1;
            printf("%d\t %s\n", OSTimeGet(), "Task1");
            if (point_t->LR1_Time < point_t->LR2_Time) {//如果這個Task是R1先做
                if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                    OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                    printf("%d\t %s\n", OSTimeGet(), "Task1 get R1");//print 題目要求
                    OSMutexPend(R1, 0, &err);
                    OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                    printf("%d\t %s\n", OSTimeGet(), "Task1 release R1");//print 題目要求
                    OSMutexPost(R1);
                    OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
                }
                if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                    OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                    printf("%d\t %s\n", OSTimeGet(), "Task1 get R2");//print 題目要求
                    OSMutexPend(R2, 0, &err);
```

```
                    if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                        OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 get R2");//print 題目要求
                        OSMutexPend(R2, 0, &err);
                        OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 release R2");//print 題目要求
                        OSMutexPost(R2);
                        OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
                    }
                }

                else {//如果這個Task是R2先做
                    if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                        OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 get R2");//print 題目要求
                        OSMutexPend(R2, 0, &err);
                        OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 release R2");//print 題目要求
                        OSMutexPost(R2);
                        OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
                    }
                    if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                        OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 get R1");//print 題目要求
                        OSMutexPend(R1, 0, &err);
                        OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task1 release R1");//print 題目要求
                        OSMutexPost(R1);
                        OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
                    }
                }
            }
            if (ptcb->OSTCBId == 2 && t2 == 0) {
                t2 = 1;
                printf("%d\t %s\n", OSTimeGet(), "Task2");//print 題目要求

                if (point_t->LR1_Time < point_t->LR2_Time) {//如果這個Task是R1先做
                    if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                        OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
                        OSMutexPend(R1, 0, &err);
                        OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
                        OSMutexPost(R1);
                        OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
                    }
                    if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                        OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                        printf("%d\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
```

```
            }
            if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
                OSMutexPend(R2, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
                OSMutexPost(R2);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
        }

        else {//如果這個Task是R2先做
            if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 get R2");//print 題目要求
                OSMutexPend(R2, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 release R2");//print 題目要求
                OSMutexPost(R2);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
            if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 get R1");//print 題目要求
                OSMutexPend(R1, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task2 release R1");//print 題目要求
                OSMutexPost(R1);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
        }
    }

    if (ptcb->OSTCBId == 3 && t3 == 0) {
        t3 = 1;
        printf("%d\t %s\n", OSTimeGet(), "Task3");//print 題目要求

        if (point_t->LR1_Time < point_t->LR2_Time) {//如果這個Task是R1先做
            if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 get R1");//print 題目要求
                OSMutexPend(R1, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 release R1");//print 題目要求
                OSMutexPost(R1);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
            if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 get R2");//print 題目要求
                OSMutexPend(R2, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 release R2");//print 題目要求
                OSMutexPost(R2);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
        }
        else {//如果這個Task是R2先做
            if (point_t->LR2_Time != 0 && point_t->UR2_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 get R2");//print 題目要求
                OSMutexPend(R2, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R2的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 release R2");//print 題目要求
                OSMutexPost(R2);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR2_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
            if (point_t->LR1_Time != 0 && point_t->UR1_Time != 0) {
                OSTimeDly_arr(point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待get R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 get R1");//print 題目要求
                OSMutexPend(R1, 0, &err);
                OSTimeDly_arr(point_t->UR1_Time - point_t->LR1_Time, OSTCBPrioTbl[2]);//讓Task等待release R1的時間
                printf("%d\t %s\n", OSTimeGet(), "Task3 release R1");//print 題目要求
                OSMutexPost(R1);
                OSTimeDly_arr(point_t->arrivaTime + point_t->executionTime - point_t->UR1_Time, OSTCBPrioTbl[2]);//讓Task等待Task結束的時間
            }
        }
    }
    ptcb = ptcb->OSTCBNext;
}

/*
▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲
*                                   PA03-NPCS
*******************************************************************************************************
*/
```

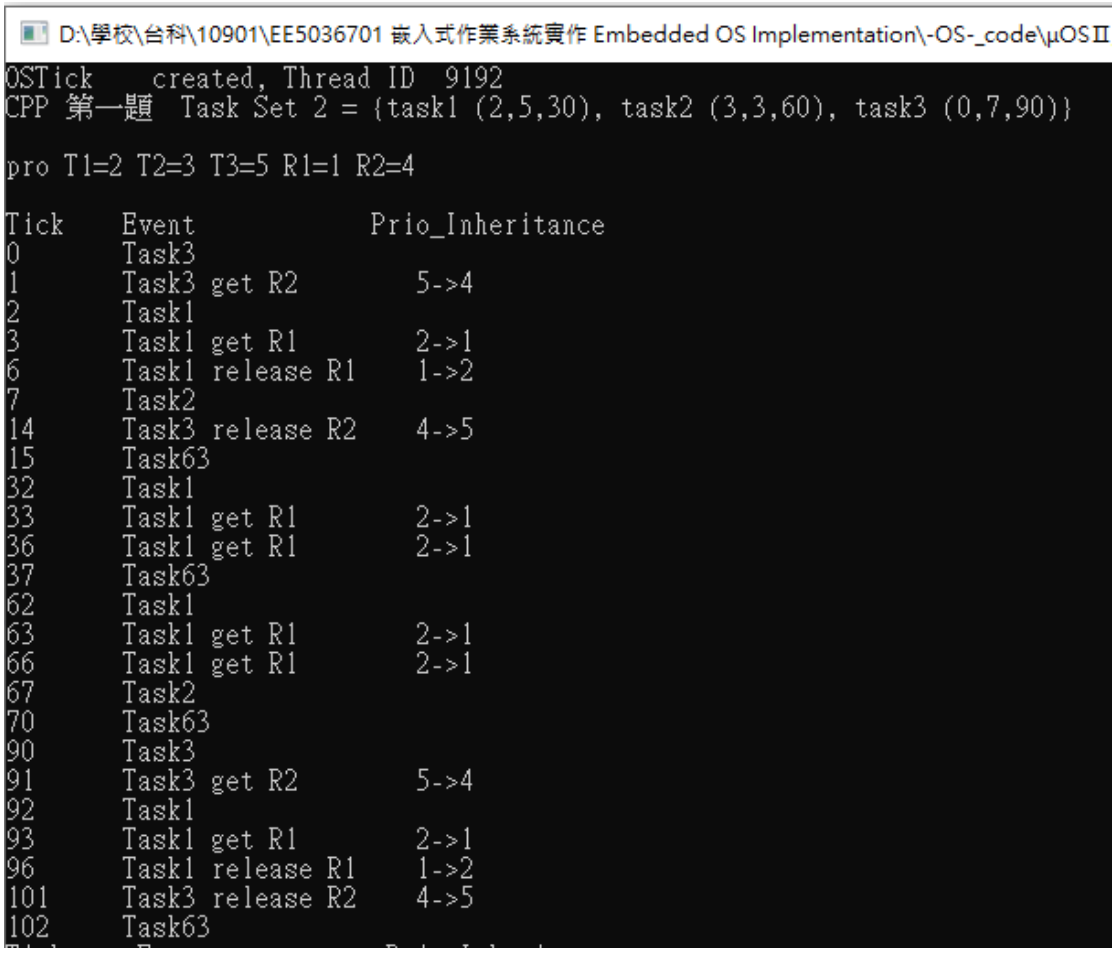
NPCS01		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	31	32	33	34	35	36	37	61	62	63	64	65	66	67	68	69	70	89	90	91	92	93	94	95	96	97	98	99	100	101	102	121	122	123	124	125	126	127	128
R1	Task1(2,5,30)			1		1		1											1		1		1			1		1		1								1		1		1								1		1		1		
R2	Task2(3,3,60)					2																						2																								2				
	Task3(0,7,90)	3			3			3																												3			3			3														
T1 ：3->L(R2) 6->U(R2)																																																								
T3：1->L(R2) 6->U(R2)		3			3			1		1		1	2	2	2	3			1		1		1			1		1		2	2	2	1			3			3			1		1		1	3			1		1		2	2	2
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	31	32	33	34	35	36	37	61	62	63	64	65	66	67	68	69	70	89	90	91	92	93	94	95	96	97	98	99	100	101	102	121	122	123	124	125	126	127	128
		T3	G2					R2	G1			R1	T2			T3	T63		T1	G1			R1	T63		T1	G1			R1			T1	T63		T3	G2					R2	G1			R1	T3	T63		T1	G1			R1		
								T1																						T2												T1												T2		

NPCS02		0	1	2	3	4	5	6	7	8	9	10	11	12	13	31	32	33	34	35	36	37	38	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	91	92	93	94	95	96	97	98	119	120	121	122	123	124	125	126	127	128
R1	Task1(2,6,30)			1	1		1		1								1	1		1		1					1	1		1		1								1	1		1		1					1	1		1		1	
R2	Task2(0,7,60)	2		2		2		2																	2		2		2		2																	2		2		2		2		
T1 ：3->L(R1) 5->L(R2) 6->U(R2 R1)																																																								
T2：1->L(R2) 4->L(R1) 6->U(R1R2)		2		2		2		1	1		1		1	2			1	1		1		1			2		2		2		1	1		1		1	2			1	1		1		1			2		2		2		1	1	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	31	32	33	34	35	36	37	38	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	91	92	93	94	95	96	97	98	119	120	121	122	123	124	125	126	127	128
		T2	G2			R2		R1	G1		R1		R2	T2	T63		T1	G1		R1		R2	T63		T2	G2			R2		R1	G1		R1		R2	T2	T63		T1	G1		R1		R2	T63		T2	G2			R2		R1	G1	
						G1		T1			G2									G2									G1		T1			G2									G2									G1		T1		

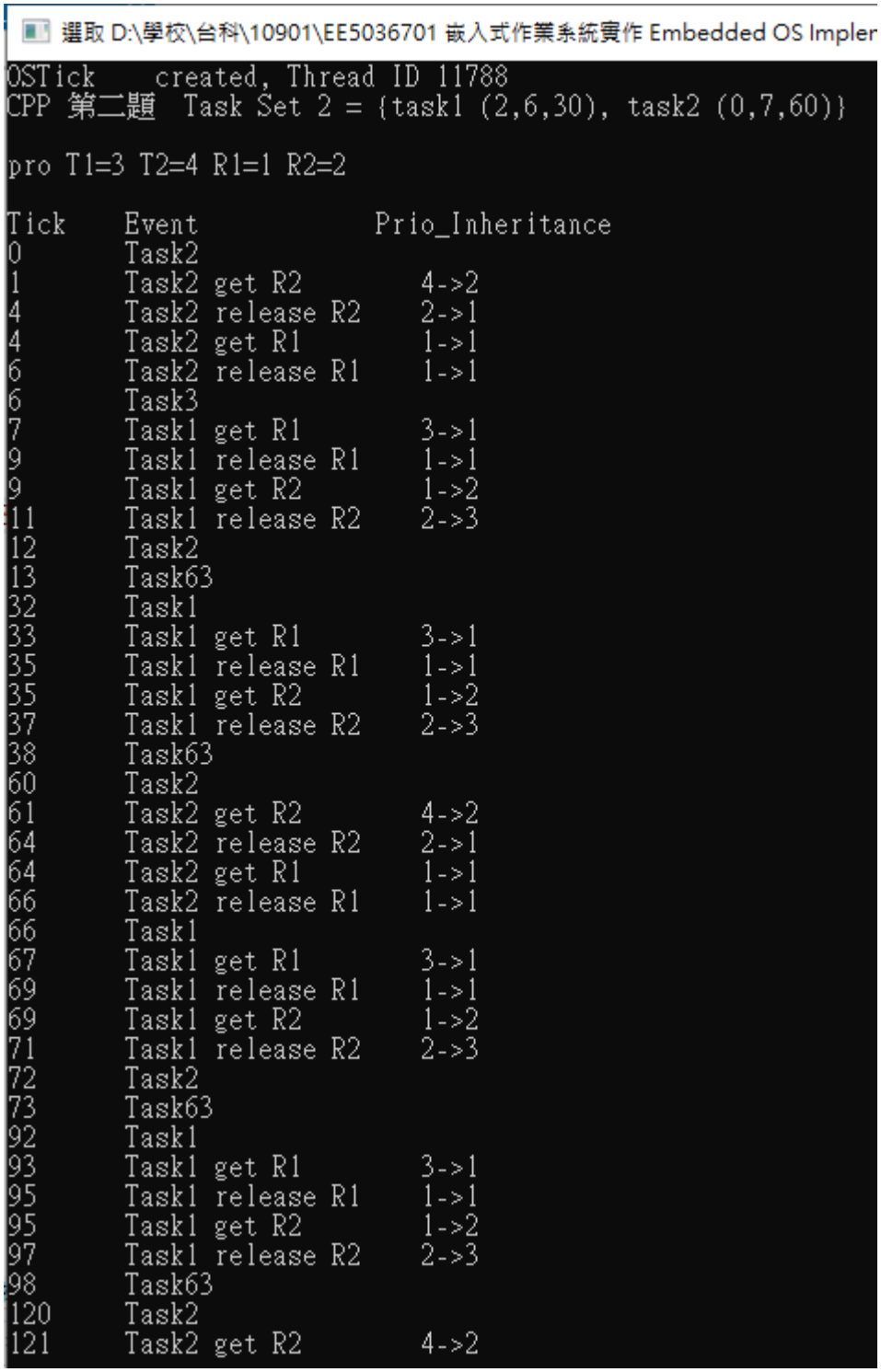
[PART II] CPP Implementation

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).

main.c* os_time.c ucos_ii.h os_core.c*

OS2 (全域範圍)

```c
****************************************************************************************************
*                                              PA03
▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼
*/
typedef struct {
    int arrivaTime;//到達時間
    int executionTime;//執行時間
    int periodTime;//週期時間
    int LR1_Time;//Lock R1 與到達時間的時間差
    int UR1_Time;//UnLock R1 與到達時間的時間差
    int LR2_Time;//Lock R2 與到達時間的時間差
    int UR2_Time;//UnLock R2 與到達時間的時間差
}TaskData;
/*
▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲
*                                              PA03
****************************************************************************************************
*/
```

```c
static TaskData Task1= {
    .arrivaTime=2,
    .executionTime=5,
    .periodTime=30,
    .LR1_Time=1,
    .UR1_Time=4,
    .LR2_Time=0,
    .UR2_Time=0
};

static TaskData Task2 = {
    .arrivaTime = 3,
    .executionTime = 3,
    .periodTime = 60,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 0,
    .UR2_Time = 0
};

static TaskData Task3 = {
    .arrivaTime = 0,
    .executionTime = 7,
    .periodTime = 90,
    .LR1_Time = 0,
    .UR1_Time = 0,
    .LR2_Time = 1,
    .UR2_Time = 6
};
```

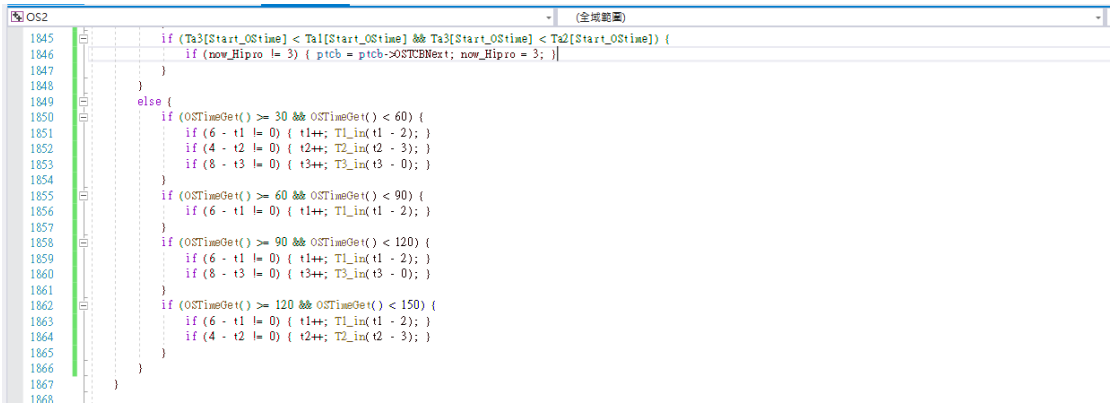
```c
#define TASK_STACKSIZE       2048


#define TASK1_PRIORITY       2
#define TASK2_PRIORITY       3
#define TASK3_PRIORITY       5

#define TASK1_ID             1
#define TASK2_ID             2
#define TASK3_ID             3

static  void  task1(void* p_arg);
static  void  task2(void* p_arg);
static  void  task3(void* p_arg);

static  OS_STK  TASK1_STK[TASK_STACKSIZE];
static  OS_STK  TASK2_STK[TASK_STACKSIZE];
static  OS_STK  TASK3_STK[TASK_STACKSIZE];
```

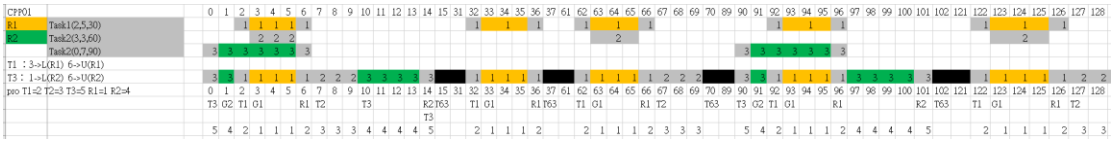
OS2 (全域範圍)

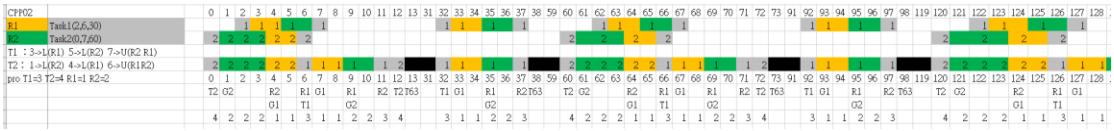
```
/*
*********************************************************************************************************
*                                                PA03-NPCS
▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼▼
*/
int Ta1[] = { 63, 63, 2, 1, 1, 1, 2 };
int Ta2[] = { 63, 63, 63, 3, 3, 3, 63 };
int Ta3[] = { 5, 4, 4, 4, 4, 4, 5 };
int now_Hipro = 0;
int Start_OStime = 0;
int t1 = 0; t2 = 0; t3 = 0;
int a1 = 5; a2 = 3; a3 = 8;
void T1_in(int c) {
    OS_TCB* ptcb;
    ptcb = OSTCBPrioTbl[OSPrioHighRdy];
    TaskData* point_t = ptcb->OSTCBExtPtr;
    if (c == 1) {printf("%d\t Task1\n", OSTimeGet());}//print 題目要求
    if (c == point_t->LR1_Time + 1) { printf("%d\t Task1 get R1       %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); }//print 題目要求
    if (c == point_t->UR1_Time + 1) { printf("%d\t Task1 release R1    %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); }//print 題目要求
    if (c == point_t->LR2_Time + 1) { printf("%d\t Task1 get R2       %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); }//print 題目要求
    if (c == point_t->UR2_Time + 1) { printf("%d\t Task1 release R2    %d->%d\n", OSTimeGet(), Ta1[Start_OStime - 1], Ta1[Start_OStime]); }//print 題目要求
}

void T2_in(int c) {
    OS_TCB* ptcb;
    ptcb = OSTCBPrioTbl[OSPrioHighRdy];
    TaskData* point_t = ptcb->OSTCBExtPtr;
    if (c == 1) { printf("%d\t Task2\n", OSTimeGet()); }//print 題目要求
    if (c == point_t->LR1_Time + 1) { printf("%d\t Task2 get R1       %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); }//print 題目要求
    if (c == point_t->UR1_Time + 1) { printf("%d\t Task2 release R1    %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); }//print 題目要求
    if (c == point_t->LR2_Time + 1) { printf("%d\t Task2 get R2       %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); }//print 題目要求
    if (c == point_t->UR2_Time + 1) { printf("%d\t Task2 release R2    %d->%d\n", OSTimeGet(), Ta2[Start_OStime - 1], Ta2[Start_OStime]); }//print 題目要求

}

void T3_in(int c) {
    OS_TCB* ptcb;
    ptcb = OSTCBPrioTbl[OSPrioHighRdy];
    TaskData* point_t = ptcb->OSTCBExtPtr;
    if (c == 1) { printf("%d\t Task3\n", OSTimeGet()); }//print 題目要求
    if (c == point_t->LR1_Time + 1) { printf("%d\t Task3 get R1       %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); }//print 題目要求
    if (c == point_t->UR1_Time + 1) { printf("%d\t Task3 release R1    %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); }//print 題目要求
    if (c == point_t->LR2_Time + 1) { printf("%d\t Task3 get R2       %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); }//print 題目要求
    if (c == point_t->UR2_Time + 1) { printf("%d\t Task3 release R2    %d->%d\n", OSTimeGet(), Ta3[Start_OStime - 1], Ta3[Start_OStime]); }//print 題目要求
}
```

OS2 (全域範圍)

```

#define R1_PRIO              1
#define R2_PRIO              4
OS_EVENT* R1;
OS_EVENT* R2;

static  void  OS_SchedNew (void)
{
#if OS_LOWEST_PRIO <= 63u                       /* See if we support up to 64 tasks                   */
    INT8U   y;
    y             = OSUnMapTbl[OSRdyGrp];
    OSPrioHighRdy = (INT8U)((y << 3u) + OSUnMapTbl[OSRdyTbl[y]]);

    OS_TCB* ptcb;
    INT8U err;
    R1 = OSMutexCreate(R1_PRIO, &err);
    R2 = OSMutexCreate(R2_PRIO, &err);
    if (OSPrioCur != OSPrioHighRdy) {
        ptcb = OSTCBPrioTbl[OSPrioHighRdy];
        TaskData* point_t = ptcb->OSTCBExtPtr;
        if (ptcb->OSTCBId == 65535) {
            printf("%d\t %s\n", OSTimeGet(), "Task63");//print 題目要求
            t1 = 0; t2 = 0; t3 = 0;
        }
        if (ptcb->OSTCBId == 1) {
            if (OSTimeGet() < 30) { t1++; T1_in(t1); }
            else {t1++; T1_in(t1-2); }

        }
        if (ptcb->OSTCBId == 2) {
            if (OSTimeGet() < 30) { t2++; T2_in(t2); }
            else { t2++; T2_in(t2 - 2); }
        }
        if (ptcb->OSTCBId == 3) {
            if (OSTimeGet() < 30) { t3++; T3_in(t3); }
            else { t3++; T3_in(t3 - 2); }
        }
        Start_OStime = OSTimeGet() % 30;
        if (Start_OStime <= 6) {
            if (Ta1[Start_OStime] < Ta2[Start_OStime] && Ta1[Start_OStime] < Ta3[Start_OStime]) {
                if (now_Hipro != 1) { ptcb = ptcb->OSTCBNext; now_Hipro = 1; }
            }
            if (Ta2[Start_OStime] < Ta1[Start_OStime] && Ta2[Start_OStime] < Ta3[Start_OStime]) {
                if (now_Hipro != 2) { ptcb = ptcb->OSTCBNext; now_Hipro = 2; }
            }
            if (Ta3[Start_OStime] < Ta1[Start_OStime] && Ta3[Start_OStime] < Ta2[Start_OStime]) {
                if (now_Hipro != 3) { ptcb = ptcb->OSTCBNext; now_Hipro = 3; }
```







[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.