Code Used for the assignment:

```
#include "FreeRTOS.h"
#include "task.h"
#include "uart0_min.h"

#define SCENARIO_1

void vTaskOneCode(void *p)
{
   while(1)
   {
    uart0_puts("Task_1: Says aaaaaaaaaaaaaaaaaaaa");
    vTaskDelay(100); // This sleeps the task for 100ms (because 1 RTOS tick = 1 millisecond)
   }
}

// Create another task and run this code in a while(1) loop
void vTaskTwoCode(void *p)
{
   while(1)
   {
    uart0_puts("Task_2: Says bbbbbbbbbbbbbbbbbbbb");
    vTaskDelay(100);
   }
}

// You can comment out the sample code of lpc1758_freertos project and run this code instead
int main(int argc, char const *argv[])
{
   /// This "stack" memory is enough for each task to run properly (512 * 32-bit) = 2Kbytes stack
   const uint32_t STACK_SIZE_WORDS = 512;



#ifdef SCENARIO_1  // 1) Same Priority: A = 1, B = 1
   xTaskCreate(vTaskOneCode, "task_1",
            STACK_SIZE_WORDS, NULL, PRIORITY_LOW, NULL);
   xTaskCreate(vTaskTwoCode, "task_2",
            STACK_SIZE_WORDS, NULL, PRIORITY_LOW, NULL);
#endif // SCENARIO_1



#ifdef SCENARIO_2 // 2) Different Priority: A = 2, B = 1
   xTaskCreate(vTaskOneCode, "task_1",
            STACK_SIZE_WORDS, NULL, PRIORITY_LOW, NULL);
   xTaskCreate(vTaskTwoCode, "task_2",
            STACK_SIZE_WORDS, NULL, PRIORITY_MEDIUM, NULL);
#endif // SCENARIO_2
```

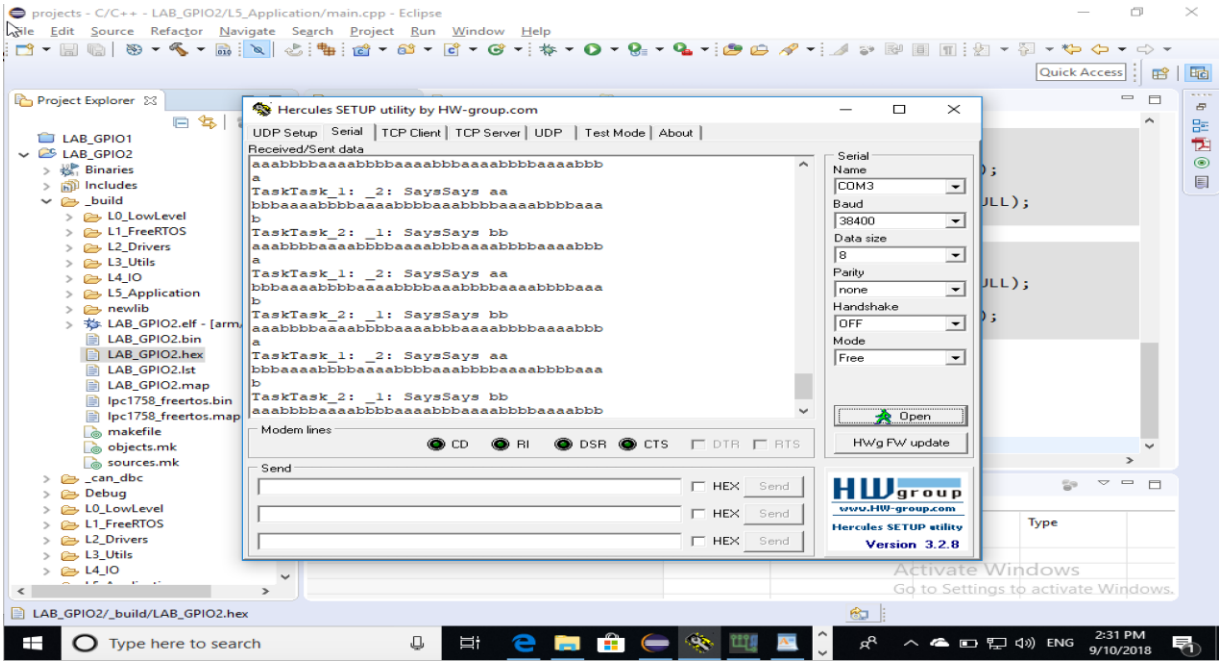
```
#ifdef SCENARIO_3 // 3) Different Priority: A = 2, B = 1
    xTaskCreate(vTaskOneCode, "task_1",
                STACK_SIZE_WORDS, NULL, PRIORITY_MEDIUM, NULL);
    xTaskCreate(vTaskTwoCode, "task_2",
                STACK_SIZE_WORDS, NULL, PRIORITY_LOW, NULL);
#endif // SCENARIO_3

    /* Start Scheduler */
    vTaskStartScheduler();

    return 0;
}
```

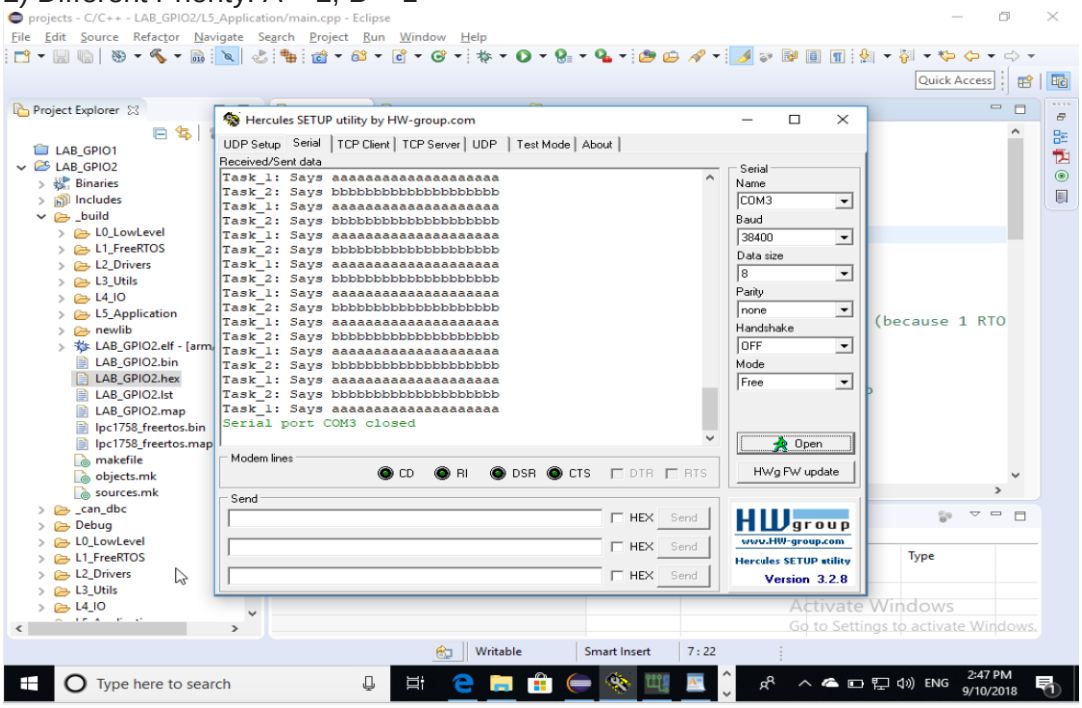
Scenario Execution Screen Shots:

1) Same Priority: A = 1, B = 1



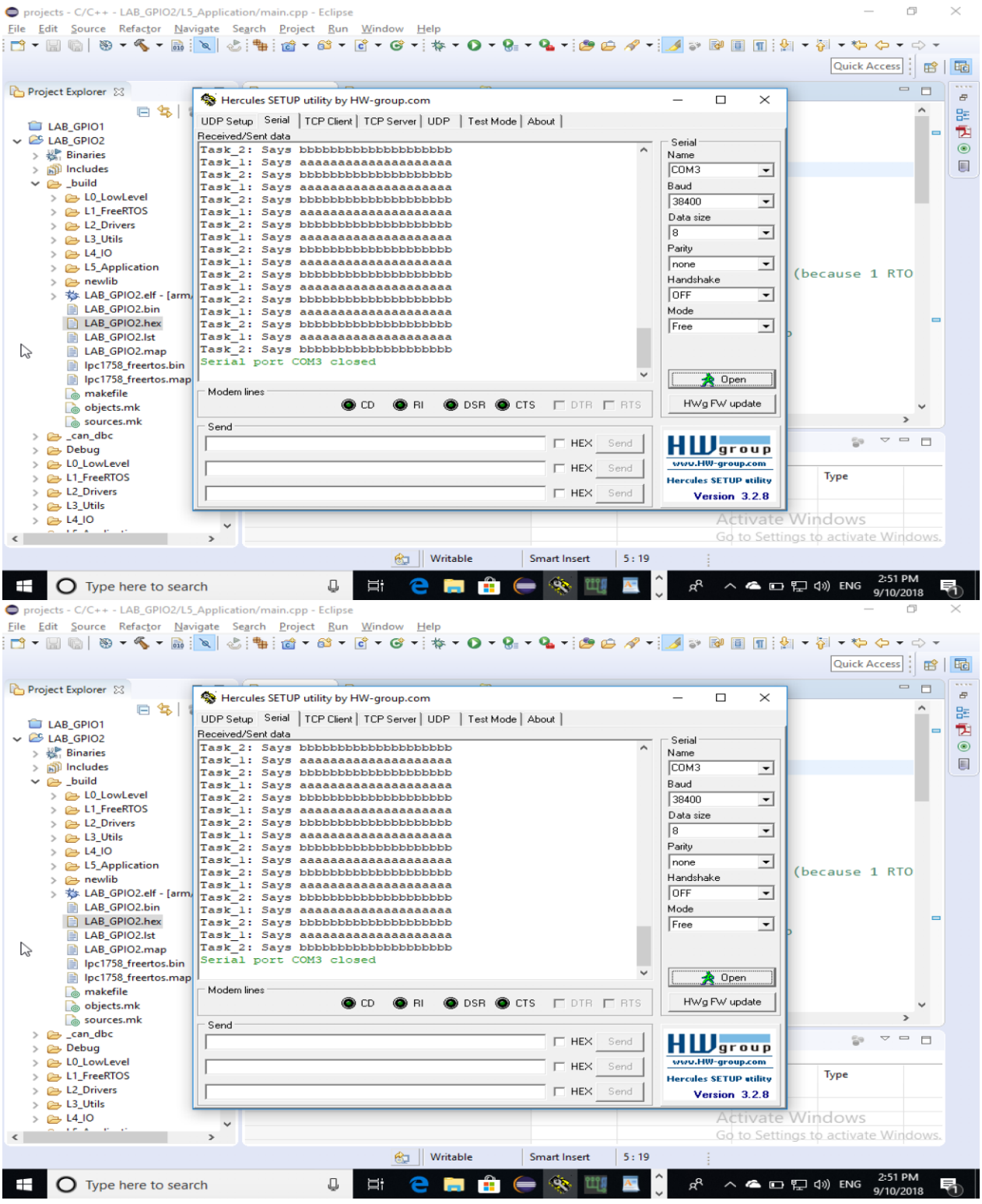
Explanation: As both Task_1 and Task_2 have the same priority and there is no synchronization mechanism in place to handle the race condition between these tasks, we are getting mixed up output.

2) Different Priority: A = 2, B = 1



Explanation: As Task_1 has higher priority than Task_2 so Task_1 executes first and goes into the waiting state due to a call to vTaskDelay() of 100 clock cycles. Once the scheduler finds that there is no ready or running task with a higher priority, it gives a chance for the lower priority Task_2 to execute.

3) Different Priority: A = 1, B = 2



Explanation: In this scenario, Task_2 has higher priority than Task_1 so Task_2 executes first and goes into the waiting state due to a call to vTaskDelay() of 100 clock cycles. Once the scheduler finds that there is no ready or running task with a higher priority, it gives a chance for the lower priority Task_1 to execute.