The Problem Statement for Q2

You must answer the following two questions to the best of your ability.

If you are not sure of the answer to a question, try to answer it anyways. It is still better to answer something (and show you tried) than nothing.

If something seems unclear or ambiguous, make some reasonable assumptions and state them in your answer.
Feel free to use any documentation you want.

Question 2:

You have an single-core STM32 microcontroller (MCU) with the following setup:
- The MCU is programmed with the code in file "question2.cpp" (assume that any other necessary file (e.g. headers) are available, and that the code compiles and runs without issues (e.g. heap and stack are always large enough)).
- The code uses FreeRTOS, a real-time operating system (RTOS) which uses priority-based preemptive scheduling.
  For detailed information about the FreeRTOS functions used in the code, please consult the adjoined "FreeRTOS_Reference_Manual_V10.0.0.pdf" document.
- A GPIO pin called NOTIF is connected to an external digital signal.
- A UART output is connected to your terminal, allowing you to see what the MCU prints.

Answer the following sub-questions (each one can be answered without considering the others):
a) The NOTIF signal is low from the beginning. You start the MCU and wait 5 seconds, during which the NOTIF signal doesn't change.
  Assume that variable "my_value" is equal to 12 at the start. What exactly will be printed to your terminal? Justify your answer.

b) The NOTIF signal is low from the beginning. The executions of tasks A and B are currently waiting on lines 39 (ulTaskNotifyTake) and 54 (xQueueReceive), respectively. Suddenly, the NOTIF signal goes from low to high. Then, we wait 5 seconds.
  Assume that variable "my_value" is equal to 12 at the start. What exactly will be printed to your terminal? Justify your answer.

c) The NOTIF signal is high from the beginning. The executions of tasks A and B are currently waiting on lines 39 (ulTaskNotifyTake) and 54 (xQueueReceive), respectively. Suddenly, the NOTIF signal goes from high to low. Then, we wait 5 seconds.
  Assume that variable "my_value" is equal to 24 at the start. What exactly will be printed to your terminal? Justify your answer.

Developing the Solution for Q2

Step 1:

Analysis of the program/code question2.cpp

Illustration of the code

```
#include <stdio.h>

// Constant definitions and HAL functions declarations
#include "main.h"

// FreeRTOS-related headers
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "queue.h"

static volatile int my_value = INIT_VALUE; // see instructions.txt to know what numerical value this corresponds to
static volatile TaskHandle_t my_task;
static volatile QueueHandle_t my_queue;
static volatile SemaphoreHandle_t my_semaphore;

// EXTI8 interrupt handler
extern "C" void EXTI8_IRQHandler()
{
  BaseType_t woken = pdFALSE;
  if (__HAL_GPIO_EXTI_GET_FALLING_IT(NOTIF_Pin)){
    __HAL_GPIO_EXTI_CLEAR_FALLING_IT(NOTIF_Pin);
    my_value /= 12;
    xQueueSendFromISR(my_queue, (int*)&my_value, &woken);
    my_value = 0;
  } else if (__HAL_GPIO_EXTI_GET_RISING_IT(NOTIF_Pin)){
    __HAL_GPIO_EXTI_CLEAR_RISING_IT(NOTIF_Pin);
    my_value = 41;
    vTaskNotifyGiveFromISR((TaskHandle_t)my_task, &woken);
    my_value = 42;
  }
  portYIELD_FROM_ISR(woken);
}

static void task_A(void* arg)
{
  printf("Task A start\r\n");
  for (;;){
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY);
    int event = my_value;
    printf("hello %d\r\n", event);
    xQueueSend(my_queue, &event, portMAX_DELAY);
    printf("how are you?\r\n");
```

```
        xSemaphoreTake(my_semaphore, portMAX_DELAY);
        printf("goodbye\r\n");
    }
}

static void task_B(void* arg)
{
    printf("Task B start\r\n");
    for (;;){
        int event;
        xQueueReceive(my_queue, &event, portMAX_DELAY);
        printf("got %d\r\n", event);
        switch (event){
        break; case 0:
            printf("impressive\r\n");
            vTaskSuspend(nullptr);
            printf("very nice\r\n");
            xSemaphoreGive(my_semaphore);
        break; case 1:
            printf("not impressive\r\n");
            vTaskSuspend(nullptr);
            xSemaphoreGive(my_semaphore);
            printf("not very nice\r\n");
        break; case 41:
            xSemaphoreGive(my_semaphore);
            printf("great!\r\n");
        break; case 42:
            xSemaphoreGive(my_semaphore);
            printf("the answer\r\n");
        break; default:
            event /= 12;
            vTaskNotifyGive((TaskHandle_t)my_task);
            xQueueSend(my_queue, &event, portMAX_DELAY);
            printf("not the answer\r\n");
        }
        ++my_value;
    }
}

extern void SystemClock_Config();

static void MX_GPIO_Init()
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    __HAL_RCC_GPIOF_CLK_ENABLE();
    GPIO_InitStruct.Pin = NOTIF_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING; // External Interrupt Mode with
Rising/Falling edge trigger detection
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
    // Enable the interrupt used by the NOTIF pin
    HAL_NVIC_SetPriority(EXTI8_IRQn, 2, 0);
```

```
   HAL_NVIC_EnableIRQ(EXTI8_IRQn);
}

extern void MX_USART2_UART_Init();

int main()
{
   // Reset all peripherals, initialize the flash interface and the systick
   HAL_Init();
   // Configure the system clock
   SystemClock_Config();
   // Initialize all GPIOs
   MX_GPIO_Init();
   // Initialize the UART peripheral used by printf for printing to the user's terminal
   // Note: Printing is a blocking operation. Interrupts, DMA or FreeRTOS function calls are not used for
printing.
   MX_USART2_UART_Init();

   printf("Ready?\r\n");
   my_queue = xQueueCreate(16, sizeof(int));
   my_semaphore = xSemaphoreCreateBinary();
   xTaskCreate(task_A, "Task A", 1024, nullptr, 6, (TaskHandle_t*)&my_task);
   xTaskCreate(task_B, "Task B", 1024, nullptr, 17, nullptr);
   printf("Start!\r\n");
   vTaskStartScheduler();
   printf("what is this I don't even\r\n");
}
```

Challenge:

Please excuse me for Q2. I am not an expert in RTOS and I don't have any hands on with FreeRTOS as of Yet. However never the less, I am extremely interested in embedded systems and want to plan my career accordingly.

The Solution:

a) The NOTIF signal is low from the beginning. You start the MCU and wait 5 seconds, during which the NOTIF signal doesn't change. Assume that variable "my_value" is equal to 12 at the start. What exactly will be printed to your terminal? Justify your answer.

Answer #a: *"Nothing accept "**Start**" and "**What is this I dont't even**" from the main function shall be printed". The reason is that there was not external interrupt yet in so task_A and task_B are still waiting at lines 39 and 54 respectively.*

b) The NOTIF signal is low from the beginning. The executions of tasks A and B are currently waiting on lines 39 (ulTaskNotifyTake) and 54 (xQueueReceive), respectively. Suddenly, the NOTIF signal goes from low to high. Then, we wait 5 seconds.
Assume that variable "my_value" is equal to 12 at the start. What exactly will be printed to your terminal? Justify your answer.

Answer #b: *task_A shall execute and print "***hello***" and "***how are you***" lines after that the default case in task_B will execute and print* ***"not the answer" and then "not impressive", and" not very nice".***

Reason since the my_value is 12. Therefore initially the default case in task_B executes and then 12/12 becomes 1 so it will execute the case 1 before getting suspended..

c) The NOTIF signal is high from the beginning. The executions of tasks A and B are currently waiting on lines 39 (ulTaskNotifyTake) and 54 (xQueueReceive), respectively. Suddenly, the NOTIF signal goes from high to low. Then, we wait 5 seconds.

Assume that variable "my_value" is equal to 24 at the start. What exactly will be printed to your terminal? Justify your answer.

Answer #c: *task_A shall execute and print "***hello***" and "***how are you***" lines. after that the default case in task_B will execute and prints many times* ***"not the answer", and once "not impressive", and" not very nice"***

Reason since the value is 24, with falling signal as per the IRQ Handler takes the value 24 and divides it by12 making it 2 and there is no case for event 2 in task_B. So the default gets executed 10 times until the value gets incremented to 12 again and finally executes case 1 before getting suspended.