1. What are the GPIO control registers that the lab mentions? Briefly describes each of their functions.
    - GPIO port mode register (GPIOX_MODER): configures the I/O mode of a pin, e.g input mode, general purpose output mode, analog mode.
    - GPIO port output type register (GPIOX_OTYPER): configures the I/O output type.
    - GPIO port output speed register (GPIOX_OSPEEDR): configures the I/O output speed.
    - GPIO port pull-up/pull-down register (GPIOX_PUPDR): configures the I/O pull-up or pull-down.
    - GPIO port input data register (GPIOX_IDR): stores the input data.
    - GPIO port output data register (GPIOX_ODR): output data that can be read and written by the software.
    - GPIO port bit set/reset register (GPIOX_BSSR): Write-only register which sets or clears the ODR register.
    - GPIO port configuration lock register (GPIOX_LCKR): Locks the configurations registers for a pin.
    - GPIO port bit reset register (GPIOX_BRR): Similar to BSSR but just to clear bits.
    - GPIO alternate function high/low registers (GPIOX_AFRL/GPIOX_AFRH): Configures alternate function to each pin.
2. What values would you want to write to the bits controlling a pin in the GPIOx_MODER register in order to set it to analog mode?
    - 11

3. Examine the bit descriptions in GPIOx_BSRR register: which bit would you want to set to clear the fourth bit in the ODR?
    - Set the 19th bit to reset ODR3, counting from 1.
4. Perform the following bitwise operations:
    - 0xAD | 0xC7 = 0xEF
    - 0xAD & 0xC7 = 0x85
    - 0xAD & ~(0xC7) = 0x28
    - 0xAD ^ 0xC& = 0x6A
5. How would you clear the 5th and 6th bits in a register while leaving the other's alone?
    - If the register has 8 bits, to clear the 5th and 6th bit is to do an AND bitwise operator with the register and `~((1 << 5) | (1 << 6))`
    - Example: `Register &= ~((1 << 5) | (1 << 6))`
6. What is the maximum speed the STM32F072R8 GPIO pins can handle in the lowest speed setting?
    - 2MHz or 500 ns
7. What RCC register would you manipulate to enable the following peripherals: (use the comments next to the bit defines for better peripheral descriptions)
    - TIM1 (TIMER1): Register 11
        - RCC->APB2ENR |=RCC_APB2ENR_TIM1EN;
    - DMA1: Register 0
        - RCC->AHBENR |= RCC_AHBENR_DMA1EN;
    - I2C1: Register 21
        - RCC->APB1ENR |= RCC_APB1ENR_12C1EN;