```
**********************
**                  **
**    QUESTION a    **
**                  **
**********************
```

Initialize(): This function takes a pointer to an int. It first dereferences the pointer and stores the file descriptor of phyiscal memory "dev/mem/". Next, it maps the physical memory to virtual memory and returns the address of the virtual memory

Finalize(): This function unmaps the virtual memory and closes the file descriptor

RegisterRead(): This function takes a memory address and an offset. It reads from memory at (address + offset) and returns the stored value.

RegisterWrite():This function takes a memory address, an offset, and a value. It then stores the given value at (address + offset)

```
****************************
**                        **
**     QUESTION b & c     **
**                        **
****************************
```

** CODE **

```
/*
 *    Project:   Lab 03 Pre-Lab
 *    Author:          Matthew Springer
 *    Date:      January 31, 2017
 */

#include <iostream>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
using namespace std;

// Physical base address of GPIO
const unsigned gpio_address = 0x400d0000;
```

```
// Length of memory-mapped IO window
const unsigned gpio_size = 0xff;

const int gpio_led1_offset = 0x12C;  // Offset for LED1
const int gpio_led2_offset = 0x130;  // Offset for LED2
const int gpio_led3_offset = 0x134;  // Offset for LED3
const int gpio_led4_offset = 0x138;  // Offset for LED4
const int gpio_led5_offset = 0x13C;  // Offset for LED5
const int gpio_led6_offset = 0x140;  // Offset for LED6
const int gpio_led7_offset = 0x144;  // Offset for LED7
const int gpio_led8_offset = 0x148;  // Offset for LED8

const int gpio_sw1_offset = 0x14C;  // Offset for Switch 1
const int gpio_sw2_offset = 0x150;  // Offset for Switch 2
const int gpio_sw3_offset = 0x154;  // Offset for Switch 3
const int gpio_sw4_offset = 0x158;  // Offset for Switch 4
const int gpio_sw5_offset = 0x15C;  // Offset for Switch 5
const int gpio_sw6_offset = 0x160;  // Offset for Switch 6
const int gpio_sw7_offset = 0x164;  // Offset for Switch 7
const int gpio_sw8_offset = 0x168;  // Offset for Switch 8

const int gpio_pbtnl_offset = 0x16C;  // Offset for left push button
const int gpio_pbtnr_offset = 0x170;  // Offset for right push button
const int gpio_pbtnu_offset = 0x174;  // Offset for up push button
const int gpio_pbtnd_offset = 0x178;  // Offset for down push button
const int gpio_pbtnc_offset = 0x17C;  // Offset for center push button

/**
 * Write a 4-byte value at the specified general-purpose I/O location.
 *
 * @param pBase      Base address returned by 'mmap'.
 * @param offsetOffset where device is mapped.
 * @param value      Value to be written.
 */
void RegisterWrite(char *pBase, int offset, int value)
{
     * (int *) (pBase + offset) = value;
}

/**
 * Read a 4-byte value from the specified general-purpose I/O location.
 *
 * @param pBase      Base address returned by 'mmap'.
 * @param offsetOffset where device is mapped.
 * @return           Value read.
 */
int RegisterRead(char *pBase, int offset)
{
     return * (int *) (pBase + offset);
}
```

```
/**
 * Initialize general-purpose I/O
 *  - Opens access to physical memory /dev/mem
 *  - Maps memory at offset 'gpio_address' into virtual address space
 *
 * @param fd    File descriptor passed by reference, where the result
 *              of function 'open' will be stored.
 * @return      Address to virtual memory which is mapped to physical,
 *              or MAP_FAILED on error.
 */
char *Initialize(int *fd)
{
     *fd = open( "/dev/mem", O_RDWR);
     return (char *) mmap(NULL, gpio_size, PROT_READ | PROT_WRITE,
MAP_SHARED,
                *fd, gpio_address);
}

/**
 * Close general-purpose I/O.
 *
 * @param pBase Virtual address where I/O was mapped.
 * @param fd    File descriptor previously returned by 'open'.
 */
void Finalize(char *pBase, int fd)
{
     munmap(pBase, gpio_size);
     close(fd);
}

/** Changes the state of an LED (ON or OFF)
 * @param  pBase base address of I/O
 * @param  ledNum     LED number (0 to 7)
 * @param  state
State to change to (ON or OFF)
*/
void Write1Led(char *pBase, int ledNum, int state) {
     int ledOffset = 0x12C + (ledNum * 0x004);
     RegisterWrite(pBase, ledOffset, state);
}

/** Reads the value of a switch
 * - Uses base address of I/O
 * @param  pBasebase  address of I/O
 * @param  switchNum  Switch number (0 to 7)
 * @return Switch          value read
*/
int Read1Switch(char *pBase, int switchNum) {
     int switchOffset = 0x14C + (switchNum * 0x004);
     return RegisterRead(pBase, switchOffset);
}
```

```
/**
 * Main function to interact with I/O Interfaces
 */
int main()
{
      // Initialize
      int fd;
      char *pBase = Initialize(&fd);

      // Check error
      if (pBase == MAP_FAILED)
      {
            cerr << "Mapping I/O memory failed - Did you run with
'sudo'?\n";
            exit(1); // Returns 1 to the operating system;
      }

      // ************** Put your code here **********************




      // Done
      Finalize(pBase, fd);
}
```