FALL 2025 EEL4742 – *Embedded Systems* - Quiz 1 Solution - Time: 25 Minutes

1. (2 points) Complete the code so that pressing S1 toggles the red LED on/off (not just while holding it):

```
#include <msp430fr6989.h>
#define redLED BIT0
#define BUT1   BIT1

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    P1DIR |= redLED;
    P1OUT &= ~redLED;

    P1DIR &= ~BUT1;
    P1REN |= BUT1;
    P1OUT |= BUT1;

    for(;;) {
        if((P1IN & BUT1) == 0) {
            ____________________;
            for(int i=0;i<50000;i++);
        }
    }
}
```

Answer: Fill in with toggling the LED when it is pressed.

```
// Filled line:
P1OUT ^= redLED;   // Toggle LED once per press (active-low)
```

2. (3 points) Given this snippet:

```
P1DIR |= BIT0;
P1OUT |= BIT0;
P1OUT ^= BIT0;
```

   (a) After execution, is the LED on or off? Explain why.
   Answer: OFF. The sequence sets bit 0 high (`|= BIT0`), then XORs bit 0, which flips it back to 0. Other bits are preserved.

   (b) Rewrite the code so it blinks every 1 second. Assume that the clock frequency is default 1 MHz.
   Answer: For loop for needed delay will be added.

```
// assume default ~1 MHz; tune loop as needed
volatile uint32_t i;
P1DIR |= BIT0;
P1OUT &= ~BIT0;
for(;;) {
    P1OUT ^= BIT0;               // toggle
    for(i = 0; i < 999,999; i++) {}}
```

3. (5 points) Complete the code so that:

   - Red LED (P1.0) stays on until S1 (P1.1) is NOT pressed.
   - Once S1 is pressed just once, the LED turns off and stays off.

```
#include <msp430fr6989.h>
#define RED  BIT0
#define S1   BIT1

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    // LED
    P1DIR |= RED;
    P1OUT |= RED;                          // start ON

    // Button S1 input with pull-up (active-low)
    P1DIR &= ~S1;
    P1REN |= S1;
    P1OUT |= S1;
```

```
for(;;) {
    _____________________ {
        ___________
        ___________
        ___________ }}
```

Answer: We need a variable to capture if the pin is ever pressed.

```
unsigned char latchedOff = 0;          // 0 = not pressed yet
for(;;) {
    if(!latchedOff && ((P1IN & S1) == 0)) {
        // first press -> latch OFF
        P1OUT &= ~RED;
        latchedOff = 1;
    }
    // once latchedOff==1, LED remains off forever (no further code needed)
}
```

4. (5 points) **Bonus**
Complete the code so that:

   - Red LED (P1.0) lights up when S1 (P1.1) is pressed.
   - Green LED (P9.7) lights up when S2 (P1.2) is pressed.
   - If both buttons are pressed, both LEDs turn off.

```
#include <msp430fr6989.h>
#define RED    BIT0     // P1.0
#define GREEN  BIT7     // P9.7
#define S1     BIT1     // P1.1
#define S2     BIT2     // P1.2

void main(void) {
    WDTCTL = WDTPW | WDTHOLD;
    PM5CTL0 &= ~LOCKLPM5;

    // LEDs
    P1DIR |= RED;   P1OUT &= ~RED;
    P9DIR |= GREEN; P9OUT &= ~GREEN;

    // Buttons S1, S2 as inputs with pull-ups (active-low)
    P1DIR &= ~(S1 | S2);
    P1REN |=  (S1 | S2);
    P1OUT |=  (S1 | S2);

    for(;;) {
        ____________________
        ____________________
        ____________________
        ____________________
    }}
```

Answer:

```
for(;;) {
    unsigned char s1_pressed = ((P1IN & S1) == 0);
    unsigned char s2_pressed = ((P1IN & S2) == 0);

    if(s1_pressed && s2_pressed) {
        // both pressed -> both off
        P1OUT &= ~RED;
        P9OUT &= ~GREEN;
    } else {
        // independent behavior
        if(s1_pressed) P1OUT |= RED; else P1OUT &= ~RED;
        if(s2_pressed) P9OUT |= GREEN; else P9OUT &= ~GREEN;
    }}
```