```c
#include <msp430.h>
#include <stdbool.h>
#include <stdint.h>


    uint8_t counter = 1;             // counts which itteration we are on

void led_Init(void) {
         P1DIR |= BIT0 | BIT1;               // Sets P1.0 and P1.1 as output (LED1 and LED2)
         P1OUT &= ~(BIT0 | BIT1);    // Turns LEDs off
}

void led_Blink(led_1or2) {
         if (led_1or2 == 0) {
                  P1OUT |= BIT0;                        // Blink LED1
                  __delay_cycles(10000);
                  P1OUT &= ~BIT0;
         } else {
                  P1OUT |= BIT1;                        // Blink LED2
                  __delay_cycles(10000);
                  P1OUT &= ~BIT1;
         }
}

void reset_Lock(void) {
         counter = 1;
         P1OUT |= BIT0 + BIT1;
         __delay_cycles(10000);                        // flash both LEDs to let you know you done goofed boy
         P1OUT &= ~(BIT0 + BIT1);
}

void joystick_Init(void) {
         P2DIR &= ~(BIT1 | BIT2 | BIT3 | BIT4 | BIT5);  // Sets up joystick as input
         // P2.1 - LEFT, P2.2 - RIGHT, P2.3 - CENTER, P2.4 - UP, P2.5 - DOWN
         P2REN |= BIT1 | BIT2 | BIT3 | BIT4 | BIT5;
         P2OUT |= BIT1 | BIT2 | BIT3 | BIT4 | BIT5;
}

void button_Init(void) {
         P2DIR &= ~(BIT6 | BIT7);    // Init P2.6 and P2.7 as inputs
}

// main.c
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;        // Stop watchdog timer

    led_Init();
    joystick_Init();
    button_Init();

    // The LOCK is UP DOWN UP DOWN LEFT RIGHT LEFT RIGHT

    while (1) {

         // if lock is correct

                  if (P2IN == 0xEF && counter == 1) { // Up and counter is on 1st entry

                           led_Blink(0);
                           counter = 2;                                          // blink LED and move to next entry

                           while (P2IN != 0xFF) {};             // wait until no user input

                  } if (P2IN == 0xDF && counter == 2) { // Down and counter is on 2nd entry

                           led_Blink(0);
                           counter = 3;                                          // blink LED and move to next entry

                           while (P2IN != 0xFF) {};             // wait until no user input

                  } if (P2IN == 0xEF && counter == 3) { // Up and counter is on 3rd entry

                           led_Blink(0);
                           counter = 4;                                          // blink LED and move to next entry

                           while (P2IN != 0xFF) {};             // wait until no user input

                  } if (P2IN == 0xDF && counter == 4) { // Down and counter is on 4th entry
```

```
                led_Blink(0);
                counter = 5;                                    // blink LED and move to next entry

                while (P2IN != 0xFF) {};            // wait until no user input

            } if (P2IN == 0xFD && counter == 5) { // Left and counter is on 5th entry

                led_Blink(0);
                counter = 6;                                    // blink LED and move to next entry

                while (P2IN != 0xFF) {};            // wait until no user input

        } if (P2IN == 0xFB && counter == 6) { // Right and counter is on 6th entry

            led_Blink(0);
            counter = 7;                                        // blink LED and move to next entry

            while (P2IN != 0xFF) {};            // wait until no user input

        } if (P2IN == 0xFD && counter == 7) { // Left and counter is on 7th entry

            led_Blink(0);
            counter = 8;                                        // blink LED and move to next entry

            while (P2IN != 0xFF) {};            // wait until no user input

        } if (P2IN == 0xFB && counter == 8) { // Right and counter is on 8th entry

            P1OUT |= BIT1;                                      // turn on LED 2
            __delay_cycles(20000);                      // keep LED 2 on
            counter = 1;                                        // reset counter
            while (P2IN == 0xFF) {};            // wait for user input on joystick
            P1OUT &= ~BIT1;                                     // turn off LED 2

        // if lock is wrong

        } if (counter == 1 && !(P2IN == 0xEF || P2IN == 0xFF)) {        // if 1st entry is wrong

            reset_Lock();                       // resets Lock

            } if (counter == 2 && !(P2IN == 0xDF || P2IN == 0xFF)) {        // if 2nd entry is wrong

                reset_Lock();                       // resets Lock

        } if (counter == 3 && !(P2IN == 0xEF || P2IN == 0xFF)) {        // if 3rd entry is wrong

            reset_Lock();                       // resets Lock

        } if (counter == 4 && !(P2IN == 0xDF || P2IN == 0xFF)) {        // if 4th entry is wrong

            reset_Lock();                       // resets Lock

        } if (counter == 5 && !(P2IN == 0xFD || P2IN == 0xFF)) {        // if 5th entry is wrong

            reset_Lock();                       // resets Lock

        } if (counter == 6 && !(P2IN == 0xFB || P2IN == 0xFF)) {        // if 6th entry is wrong

            reset_Lock();                       // resets Lock

        } if (counter == 7 && !(P2IN == 0xFD || P2IN == 0xFF)) {        // if 7th entry is wrong

            reset_Lock();                       // resets Lock

        } if (counter == 8 && !(P2IN == 0xFB || P2IN == 0xFF)) {        // if 8th entry is wrong

        reset_Lock();                       // resets Lock
        }
    }
}
```