```
/*
 * CE2812 - 021
 * Winter 2016
 * Lab 2 - Knight Rider Lights+
 * Name: Yahui Liang
 * Created: 12/12/2016
 */

/*
 * This project has been done once by using assembly language
 * last quarter. However, we used C language this time. Compare
 * with Java, C is harder to use. If I make a method, I should also declare
 * its signature at the top of the file (This is a little complex). Most
logic
 * in Java is about how objects interact with each other; however, in C, most
 * logic is about how numbers are being manipulated. Through comparing C with
 * assembly language, I think C is more readable. Assembly instructions are
 * very simple; therefore, we need to write many lines for one logic. In C,
 * we do not need that much code; instead, we can use less lines to represent
 * one logic. This is why this C program has less lines than the original
version.
 * However, there are many syntax problems we should be careful. Since
assembly just
 * has simple instructions, we do not needs to worry that much; however, C is
 * totally different. Pointers are things that always make me annoy.
 */

/* Files included in this program */
#include <stdio.h>
#include <inttypes.h>

/* All symbolic names */
#define F_CPU 16000000UL
#define RCC_AHB1ENR (volatile uint32_t*) 0x40023830
#define GPIOA_MODER (volatile uint32_t*) 0x40020000
#define GPIOB_MODER (volatile uint32_t*) 0x40020400
#define GPIOA_ODR (volatile uint32_t*) 0x40020014
#define GPIOB_ODR (volatile uint32_t*) 0x40020414

/* All methods' signatures */
void init_PA7_to_PA11();
void init_PB8_to_PB10_and_PB12_to_PB13();
void delay_ms(uint32_t);
void light_LED(uint16_t);
void light_LED_init();

/**
 * The program is a Knight Rider Lights+.
 */
int main(void){
      light_LED_init();
      uint16_t number = 1;
      // Light up lights by shifting bits.
      while (1) {
            for (int i = 0; i < 9; i++) {
                  light_LED(number);
                  delay_ms(200);
```

```c
                  number = number << 1;
            }
            for (int i = 0; i < 9; i++) {
                  light_LED(number);
                  delay_ms(200);
                  number = number >> 1;
            }
      }
}

/**
 * Initializes all LEDs used in this program.
 */
void light_LED_init() {
      init_PA7_to_PA11();
      init_PB8_to_PB10_and_PB12_to_PB13();
}

/**
 * The delay subroutine which delays the number of milliseconds based
 * on the argument passed in.
 */
void delay_ms(uint32_t theDelay) {
      volatile uint32_t *systick;
      systick = (uint32_t *) 0xE000E010;
      *systick = 0;
      systick[2] = 0;
      systick[0] = 0;
      systick[1] = theDelay * (F_CPU / 8000);
      systick[0] = 1; // enable the clock.
      while (!(systick[0] & (1 << 16))) {
            // nothing to do.
      }
      systick[0] = 0;
      return;
}

/**
 * Initializes the first five LEDs.
 */
void init_PA7_to_PA11() {
      /* enable the clock for GPIOA */
      *RCC_AHB1ENR |= 1;
      /* set mode */
      *GPIOA_MODER &= (~0b1111111111 << 14); // make sure bits are set to 0.
      *GPIOA_MODER |= (0b0101010101 << 14);
}

/**
 * Initializes the last five LEDs.
 */
void init_PB8_to_PB10_and_PB12_to_PB13() {
      /* enable the clock for GPIOB */
      *RCC_AHB1ENR |= (1 << 1);
      /* set mode */
      *GPIOB_MODER &= (~0b111111 << 16);
      *GPIOB_MODER &= (~0b1111 << 24);
      *GPIOB_MODER |= (0b010101 << 16);
      *GPIOB_MODER |= (0b0101 << 24);
}

/**
 * Lights up the corresponding LED based on the number passed in.
 */
void light_LED(uint16_t number) {
      uint16_t least_5_bits = (number & ~0xFFE0) << 7;
      uint16_t middle_3_bits = (number & ~0xFF1F) << 3;
      uint16_t last_2_bits = (number & ~0xFCFF) << 4;
      *GPIOA_ODR &= 0; // clear all bits to 0.
      *GPIOB_ODR &= 0;
      *GPIOA_ODR |= least_5_bits;
      *GPIOB_ODR |= middle_3_bits;
      *GPIOB_ODR |= last_2_bits;
}
```