```c
/*
 * adeja001_lab2_part1.c − April 12, 2013
 * Name: Ariana DeJaco  E−mail:adeja001@ucr.edu
 * CS Login: adeja001
 * Partner Name: Joshua DeForest−Williams E−mail jdefo002@ucr.edu
 * Lab Section: 022
 * Assignment: Lab#2 Exercise#1
 * Exercise Description: Count the number of 1s on ports A and B and output that
 number on port C
*/


#include <avr/io.h>
#include <avr/sfr_defs.h>

// Bit−access function
unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b) {
        return (b ? x | (0x01 << k) : x & ~(0x01 << k));
}
unsigned char GetBit(unsigned char x, unsigned char k) {
        return ((x & (0x01 << k)) != 0);
}

// Current Port Definitions
#define OUTPUT_DDR                      DDRC
#define OUTPUT_INPORT           PINC
#define OUTPUT_OUTPORT          PORTC
#define COUNT_A_DDR                     DDRA
#define COUNT_A_INPORT          PINA
#define COUNT_A_OUTPORT         PORTA
#define COUNT_B_DDR                     DDRB
#define COUNT_B_INPORT          PINB
#define COUNT_B_OUTPORT         PORTB

// Additional macros not defines in sfr_defs.h
//#define SET_PORT_BIT(OUTPORT, BIT)            OUTPORT |= (1 << BIT)
//#define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)

//DDRA: Configures each of port A’s physical pins to input (0) or output (1)
//PORTA: Writing to this register writes the port’s physical pins (Write only)
//PINA: Reading this register reads the values of the port’s physical pins (Read
 only)
int main(void)
{
        OUTPUT_DDR = 0xFF; //Configures port C’s 8 pins as outputs.
        OUTPUT_OUTPORT = 0x00; //Initialize output on port C to 0x00;
        COUNT_A_DDR = 0x00; // Configure Port A’s 8 pins as inputs.
        //COUNT_A_OUTPORT = 0xFF; // Configure Port A’s 8 pins as inputs.
        COUNT_B_DDR = 0x00; // Configure Port B’s 8 pins as inputs.
        //COUNT_B_OUTPORT = 0xFF; // Configure Port B’s 8 pins as inputs.

        char loop_counter;
        char count;
    while(1)
    {
       count = 0;
        for (loop_counter=0; loop_counter<8; loop_counter++)
                {
                        if (GetBit(COUNT_A_INPORT, loop_counter))
                        {
                                count +=1 ;
                        }

                        if(GetBit(COUNT_B_INPORT, loop_counter))
                        {
                                count +=1;
                        }
                        }
                OUTPUT_OUTPORT = count;
                }

        }
```

```c
/*
 * adeja001_lab2_part2.c − April 8, 2013
 * Name:  Ariana DeJaco  E−mail:adeja001@ucr.edu
 * CS Login: adeja001
 * Partner Name:Joshua DeForest−Williams E−mail jdefo002@ucr.edu
 * Lab Section: 022
 * Assignment: Lab#1 Exercise#2
 * Exercise Description: A car has a fuel−level sensor that sets PA3..PA0 to a value between 0 (empty) and 15 (full).
 * A series of LEDs connected to PC5..PC0 should light to graphically indicate the fuel level. If the fuel level is 1 or 2,
 * PC5 lights. If the level is 3 or 4, PC5 and PC4 light. Level 5−6 lights PC5..PC3. 7−9 lights PC5..PC2. 10−12
 * lights PC5..PC1. 13−15 lights PC5..PC0. Also, PC6 connects to a "Low fuel" icon, which should light if the level is 4 or less.
 * (The example below shows the display for a fuel level of 3).
*/


#include <avr/io.h>
#include <avr/sfr_defs.h>

// Bit−access function
unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b) {
        return (b ? x | (0x01 << k) : x & ~(0x01 << k));
}
unsigned char GetBit(unsigned char x, unsigned char k) {
        return ((x & (0x01 << k)) != 0);
}

// Current Port Definitions
#define LED_DDR                         DDRC
#define LED_INPORT                      PINC
#define LED_OUTPORT                     PORTC
#define SENSOR_DDR                      DDRA
#define SENSOR_INPORT           PINA
#define SENSOR_OUTPORT          PORTA

// Additional macros not defines in sfr_defs.h
#define SET_PORT_BIT(OUTPORT, BIT)              OUTPORT |= (1 << BIT)
#define CLEAR_PORT_BIT(OUTPORT, BIT)    OUTPORT &= ~(1 << BIT)

//DDRA: Configures each of port A’s physical pins to input (0) or output(1)
//PORTA: Writing to this register writes the port’s physical pins
// (Write only)
//PINA: Reading this register reads the values of the port’s physical pins
// (Read only)
int main(void)
{
   SENSOR_DDR = 0x00;
   SENSOR_OUTPORT = 0xFF; // Configure port A’s 8 pins as inputs
        LED_DDR = 0xFF;
   LED_OUTPORT = 0x00; // Configure port C’s 8 pins as outputs,
   const unsigned char Low_level = 0;
   const unsigned char Sec_low_level = 2;
   const unsigned char Mid_low_level = 4;
   const unsigned char Mid_Hi_level = 6;
   const unsigned char Sec_Hi_level = 9;
   const unsigned char High_level =12;

    while(1)
    {
       char led = 0;
       if (SENSOR_INPORT > High_level)
         {
            SET_PORT_BIT  (led,5);
         }
      if (SENSOR_INPORT > Sec_Hi_level)
         {
            SET_PORT_BIT (led,4);
         }
      if (SENSOR_INPORT > Mid_Hi_level)
         {
```

```c
            SET_PORT_BIT(led,3);
         }
      if (SENSOR_INPORT > Mid_low_level)
         {
            SET_PORT_BIT(led,2);
         }
      if (SENSOR_INPORT > Sec_low_level)
         {
            SET_PORT_BIT(led,1);
         }
      if (SENSOR_INPORT > Low_level )
         {
            SET_PORT_BIT(led,0);
         }
      if (SENSOR_INPORT <= Mid_low_level)
         {
            SET_PORT_BIT(led,6);
         }
       LED_OUTPORT = led;
      }
   }
```

```
/*
 * adeja001_lab2_part3.c − April 8, 2013
 * Name: Ariana DeJaco  E−mail:adeja001@ucr.edu
 * CS Login: adeja002
 * Partner Name: Joshua DeForest−Williams E−mail jdefo002@ucr.edu
 * Lab Section: 022
 * Assignment: Lab#1 Exercise#3
 * Exercise Description: A car has a fuel−level sensor that sets PA3..PA0
 * to a value between 0 (empty) and 15 (full). A series of LEDs connected
 * to PC5..PC0 should light to graphically indicate the fuel level.
 * If the fuel level is 1 or 2, PC5 lights. If the level is 3 or 4,
 * PC5 and PC4 light. Level 5−6 lights PC5..PC3. 7−9 lights PC5..PC2.
 * 10−12 lights PC5..PC1. 13−15 lights PC5..PC0. Also, PC6 connects to a
 * "Low fuel" icon, which should light if the level is 4 or less.
 * (The example below shows the display for a fuel level of 3).
 * In addition to the above, PA4 is 1 if a key is in the ignition,
 * PA5 is 1 if a driver is seated, and PA6 is 1 if the driver’s seatbelt
 * is fastened. PC7 should light a "Fasten seatbelt" icon if a key is in
 * the ignition, the driver is seated, but the belt is not fastened.
*/


#include <avr/io.h>
#include <avr/sfr_defs.h>

// Bit−access function
unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b) {
        return (b ? x | (0x01 << k) : x & ~(0x01 << k));
}
unsigned char GetBit(unsigned char x, unsigned char k) {
        return ((x & (0x01 << k)) != 0);
}

// Current Port Definitions
#define LED_DDR                         DDRC
#define LED_INPORT                      PINC
#define LED_OUTPORT                     PORTC
#define SENSOR_DDR                      DDRA
#define SENSOR_INPORT           PINA
#define SENSOR_OUTPORT          PORTA

// Additional macros not defines in sfr_defs.h
#define SET_PORT_BIT(OUTPORT, BIT)              OUTPORT |= (1 << BIT)
#define CLEAR_PORT_BIT(OUTPORT, BIT)    OUTPORT &= ~(1 << BIT)

//DDRA: Configures each of port A’s physical pins to input (0) or output(1)
//PORTA: Writing to this register writes the port’s physical pins
// (Write only)
//PINA: Reading this register reads the values of the port’s physical pins
// (Read only)
int main(void)
{
   SENSOR_DDR = 0x00;
   SENSOR_OUTPORT = 0xFF; // Configure port A’s 8 pins as inputs
        LED_DDR = 0xFF;
   LED_OUTPORT = 0x00; // Configure port C’s 8 pins as outputs,
   const unsigned char Low_level = 0;
   const unsigned char Sec_low_level = 2;
   const unsigned char Mid_low_level = 4;
   const unsigned char Mid_Hi_level = 6;
   const unsigned char Sec_Hi_level = 9;
   const unsigned char High_level =12;

    while(1)
    {
       char SENSOR = SENSOR_INPORT & 0x0F;
       char ignition = (SENSOR_INPORT >> 4) & 0x01;
       char seated = (SENSOR_INPORT >> 5) & 0x01;
       char seatbelt = (SENSOR_INPORT>>6)& 0x01;
       char led = 0;
       if (SENSOR > High_level)
         {
            SET_PORT_BIT  (led,0);
```

```
         }
       if (SENSOR > Sec_Hi_level)
          {
             SET_PORT_BIT (led,1);
          }
       if (SENSOR > Mid_Hi_level)
          {
             SET_PORT_BIT(led,2);
          }
       if (SENSOR > Mid_low_level)
          {
             SET_PORT_BIT(led,3);
          }
       if (SENSOR > Sec_low_level)
          {
             SET_PORT_BIT(led,4);
          }
       if (SENSOR > Low_level )
          {
             SET_PORT_BIT(led,5);
          }
       if (SENSOR <= Mid_low_level)
          {
             SET_PORT_BIT(led,6);
          }
       if (ignition && seated && !seatbelt)
          {
             SET_PORT_BIT(led, 7);
          }
        LED_OUTPORT = led;
       }
    }
```

```c
/*
 * adeja001_lab2_part4.c − April 8, 2013
 * Name: Ariana DeJaco  E−mail:adeja001@ucr.edu
 * CS Login: adeja002
 * Partner Name: Joshua DeForest−Williams E−mail jdefo002@ucr.edu
 * Lab Section: 022
 * Assignment: Lab#1 Exercise#4
 * Exercise Description: (Challenge): Read an 8−bit value on PA7..PA0
 * and write that value on PB3..PB0PC7..PC4. That is to say,
 * take the upper nibble of PINA and map it to the lower nibble of PORTB,
 * likewise take the lower nibble of PINA and map it to the upper
 * nibble of PORTC (PA7 −> PB3, PA6 −> PB2, M−^E PA1 −> PC5, PA0 −> PC4).
 */

#include <avr/io.h>
#include <avr/sfr_defs.h>

// Bit−access function
unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b) {
        return (b ? x | (0x01 << k) : x & ~(0x01 << k));
}
unsigned char GetBit(unsigned char x, unsigned char k) {
        return ((x & (0x01 << k)) != 0);
}

// Current Port Definitions
#define HIGH_NIB_DDR                    DDRC
#define HIGH_NIB_INPORT         PINC
#define HIGH_NIB_OUTPORT        PORTC
#define FULL_NIB_DDR                    DDRA
#define FULL_NIB_INPORT         PINA
#define FULL_NIB_OUTPORT        PORTA
#define LOW_NIB_DDR                     DDRB
#define LOW_NIB_INPORT          PINB
#define LOW_NIB_OUTPORT         PORTB

// Additional macros not defines in sfr_defs.h
//#define SET_PORT_BIT(OUTPORT, BIT)            OUTPORT |= (1 << BIT)
//#define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)

//DDRA: Configures each of port A’s physical pins to input (0) or output (1)
//PORTA: Writing to this register writes the port’s physical pins (Write only)
//PINA: Reading this register reads the values of the port’s physical pins (Read
 only)
int main(void)
{
        HIGH_NIB_DDR = 0xFF; //Configures port C’s 8 pins as outputs.
        HIGH_NIB_OUTPORT = 0x00; //Initialize output on port C to 0x00;
        FULL_NIB_DDR = 0x00; // Configure Port A’s 8 pins as inputs.
   FULL_NIB_OUTPORT = 0xFF; // Configure Port A’s 8 pins as inputs.
        LOW_NIB_DDR = 0x00; // Configure Port B’s 8 pins as inputs.
        LOW_NIB_OUTPORT = 0xFF; // Configure Port B’s 8 pins as inputs.
   while(1)
   {
      //PORTC (PA7 −> PB3, PA6 −> PB2, M−^E PA1 −> PC5, PA0 −> PC4).
    char temp_low_nib = (FULL_NIB_INPORT >> 4);
    LOW_NIB_OUTPORT = temp_low_nib;
    char temp_high_nib = (FULL_NIB_INPORT & 0x0f) << 4;
    HIGH_NIB_OUTPORT = temp_high_nib;
   }
}
```

```c
/*
 * adeja001_lab2_part5.c − April 8, 2013
 * Name: Ariana DeJaco  E−mail:adeja001@ucr.edu
 * CS Login: adeja001
 * Partner Name: Joshua DeForest−Williams E−mail jdefo002@ucr.edu
 * Lab Section: 022
 * Assignment: Lab#1 Exercise#5
 * Exercise Description: (Challenge): A car’s passenger−seat weight
 * sensor outputs a 9−bit value (ranging from 0 to 511) and connects
 * to input PD7..PD0PB0 on the microcontroller. If the weight is equal
 * to or above 70 pounds, the airbag should be enabled by setting PB1 to
 * 1. If the weight is above 5 but below 70, the airbag should be
 * disabled and an "Airbag disabled" icon should light by setting PB2 to
 * 1. (Neither B0 nor B1 should be set if the weight is 5 or less,
 * as there is no passenger).
 */

#include <avr/io.h>
#include <avr/sfr_defs.h>

// Bit−access function
unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b) {
        return (b ? x | (0x01 << k) : x & ~(0x01 << k));
}
unsigned char GetBit(unsigned char x, unsigned char k) {
        return ((x & (0x01 << k)) != 0);
}

// Current Port Definitions
#define WEIGHT_DDR              DDRD
#define WEIGHT_INPORT   PIND
#define WEIGHT_OUTPORT  PORTD
#define AIRBAG_DDR         DDRB
#define AIRBAG_INPORT   PINB
#define AIRBAG_OUTPORT  PORTB

// Additional macros not defines in sfr_defs.h
#define SET_PORT_BIT(OUTPORT, BIT)              OUTPORT |= (1 << BIT)
#define CLEAR_PORT_BIT(OUTPORT, BIT)    OUTPORT &= ~(1 << BIT)

//DDRA: Configures each of port A’s physical pins to input (0) or output (1)
//PORTA: Writing to this register writes the port’s physical pins (Write only)
//PINA: Reading this register reads the values of the port’s physical pins (Read
 only)
int main(void)
{
   long weight;
   char airbag = 0;
        WEIGHT_DDR = 0x00;
   WEIGHT_OUTPORT = 0xFF;
        AIRBAG_DDR = 0xFE;
   AIRBAG_OUTPORT = 0x01;
   while(1)
   {
      airbag = 0;
      weight = (WEIGHT_INPORT << 1);
      weight = weight | (AIRBAG_INPORT & 0x01);
      if (weight >= 70)
      {
         SET_PORT_BIT(airbag, 1);
      }
      else if (weight < 70 && weight >5)
      {
         CLEAR_PORT_BIT(airbag,1);
         SET_PORT_BIT(airbag, 2);
      }
      else if (weight < 5)
      {
         CLEAR_PORT_BIT(airbag,0);
         CLEAR_PORT_BIT(airbag,1);
      }
      AIRBAG_OUTPORT = airbag;

```

```c
   }
}
```