a. (2 pts) Configure INT1 for falling edge triggered interrupts, no priorities. Give bit
settings so that this interrupt is fully enabled (you cannot assume any default bit
settings).

```c
IPEN = 0;             //no priorities
INT1IF = 0;         //clear interrupt flag
INTEDG1 = 0;    //falling edge triggered
INT1IE =1;         //enable INT1 interrupt
GIE =1;               //global interrupt enable must be a ‘1’
```

b. (3 pts) Assume that INT2 has been initially configured to be rising edge triggered
and is connected to momentary push button switch. Write an ISR that will count
EVERY rising edge that occurs when the switch is pushed. Increment a variable
called `edge_count`; when the five edges have occurred disable the INT2
interrupt. Your ISR cannot assume that INT2 is the only enabled interrupt.

```c
interrupt my_isr() {
    if (INT2IF && INT2IE) {  //flag set and interrupt enabled?
        INT2IF= 0; //clear interrupt flag
        edge_count++;
        if (edge_count == 5) INT2IE = 0;   //disable interrupt after 5 edges
    }
}
```

c. (2 pts) What registers are automatically saved in the shadow registers when an
interrupt occurs? **BSR, W, STATUS**

d. (3 pts) Assume we have a circular buffer named `buf` with a maximum of 8
entries. Write a C function that waits for a character to be available, then reads the
character from the buffer. Assume the circular buffer pointers are named `tail`
and `head`; with the tail pointer used to take data out of the buffer. Your C
function must be complete (i.e, it must compile).

```c
char get_char() {
    while (head == tail);  // wait until buffer is not empty
    tail++;                 // increment the tail pointer
    if (tail == 8) tail = 0; //wrap the tail pointer
    return(buf[tail]);      //return the character from the buffer
}
```