Name: **RIN:**

Q.1 (1 pnts): What color is the car that we use in the lab?

Q.2 (2 pnts): True or **False**: An `int16_t` can hold more unique values than a `uint16_t` as it can go negative.

Q.3 (5 pnts): Two variables, `uint8_t hi, lo;`, are used to set the most significant and least significant bytes (MSB and LSB), respectively, of `uint16_t combo = 0;`. **Cross out** any commands below that **do not** successfully perform this operation.

```
combo = hi + lo;
combo = hi * 256 + lo;
combo = hi<<8 + lo;
combo = hi<<8 | lo;
combo = hi | lo>>8;
{combo += lo; combo &= hi<<8;}
```

Q.4 (9 pnts): For each expression below, determine the final value in Hexadecimal.

`0xAB | 0xBA` = _____

`0xAB || 0xBA` = _____

`0xAB | ~0xBA` = _____

`0xAB | !0xBA` = _____

`0xAB || !0xBA` = _____

`~0xAB | 0xBA` = _____

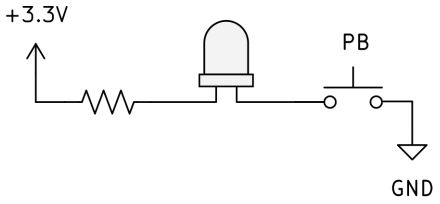
Q.5 (2 pnts): True or **False**: The following two statements are equivalent in their result.

```
P2DIR = 0xD8;     P2DIR = 0x1B<<3;
```

Q.6 (4 pnts): What will be printed on the terminal after this code segment runs? Reproduce the output *exactly*.

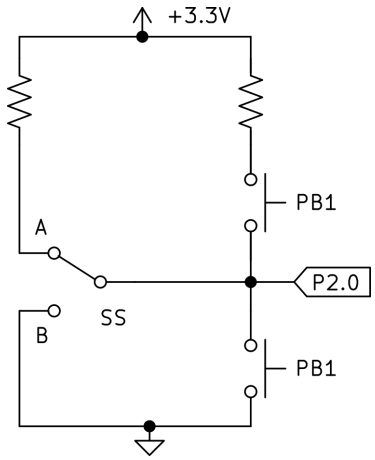
```
uint8_t a = 127;
int8_t b = -1;
printf("a = %u, b = %d, a = %x, b+1 = %u \n\r",a,b,a,b+1);
```

Q.7 (3 pnts): Consider the circuit to the right and assuming a properly sized resistor. What must happen to ensure the LED will light when the pushbutton is pressed? You may answer with a drawing or words.



Q.8 (2 pnts): If the LED above is replaced with a BiColor LED, how would your answer to the above question change?

Q.9 (4 pnts): For the circuit below, find all combinations of the switch settings such that the logic sensed by P2.0 is TRUE. Denote for each combination the state of all switches (SS: position A or B, Pushbuttons: PRESSED or UNPRESSED).



Q.10 (4 pnts): What is the final value for variable `i` after the below code segment runs?

```
uint8_t i = 0;
uint8_t j = 55;
while(i <= j){
    i += 5;
    j--;
}
```

Name: **RIN:**

Q.11 (6 pnts): Given the code segment below, indicate which pins are are known to be **inputs** and **outputs**.

```
P1DIR &= 0x30;
P1DIR |= ~0x03;
P1OUT = 0x0FA2;
```

Q.12 (6 pnts): Fill in the missing arguments for configuring a pushbutton input connected to P6.2. Additionally, draw an appropriate pushbutton circuit that would work with the initialization code.

```
GPIO_setAsInputPinWithPullDownResistor(              ,              );
```

Q.13 (6 pnts): Add a comment to each line to interpret what each line is doing. Is the code below a **Blocking** or **Non-Blocking** implementation for checking if a specific signal occurred on a GPIO pin? Assume that the `xxx` and `yyy` are the appropriate values the pins desired.

```
while(1){
    lp_cnt++;

    ovf_cntr = 0;

    while( ovf_cntr < 50 );

    GPIO_setOutputLowOnPin(yyy,yyy));

    if(!GPIO_getInputPinValue(xxx,xxx)){

        GPIO_setOutputHighOnPin(yyy,yyy));

        lp_cnt = 0;
    }
    printf("Loops since press: %u\r\n",lp_cnt);
}
```

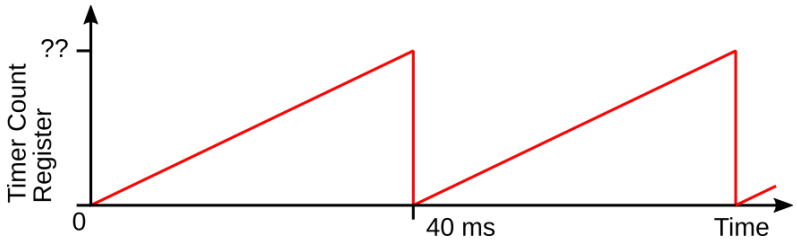
For all questions on this page: consider a generic UP counting timer with a desired 40 Hz reset frequency. The input clock divider for the timer is set to 8 and the timer count period is set to 12500.

Q.14 (6 pnts): What must the base clock frequency be (the clock source used as the input) for the described timer **and** how often does the timer increment, in seconds?

Q.15 (4 pnts): It is instead desired to have the reset frequency be 20 Hz. What two changes to the given configuration could be made to affect this change? Note that these two changes produce the change independently; only one of the changes would be necessary.

Q.16 (4 pnts): Assuming the original reset period of 40 Hz, an interrupt function for the timer is written such that the global variable `track` increments each reset. Provide line(s) of code that would reliably produce a program delay of 10 s using this support.

Q.17 (4 pnts): What numeric value for the timer is denoted by the "??" in the figure below? Give that value.



Name: **RIN:**

You may use shorthand code for the remaining problems: As long as it is clear what function/defined value you are referring to, you may shorten the name of the DriverLib functions/defined values to save time and/or space.

A simple program is desired to control one BiColor LED (**BLED: P2.0,P2.1**) and one bidirectional motor (**MOTOR ENABLE: P1.0, MOTOR DIRECTION: P1.1**), via a slideswitch (**SS: P3.0**) and one pushbutton (**PB: P3.1**). When the state of any of the inputs change, the program should hold the new state for a minimum of 1 second, measured using `Timer_A1` with a period of **25 ms**.

All input pins have external **pull-up** resistors and the motor turns on when the enable is driven **high**.

Q.18 (10 pnts): Complete the timer initialization function to initialize both the timer and timer interrupt function (bottom of page).

```
void TimerInit(){


    timer_cfg.clockSource = TIMER_A_CLOCKSOURCE_SMCLK; // 24 MHz









}

void Timer_Int(){
    Timer_A_clearInterruptFlag(                    );
    timer_counter++;
}
```

Name: **RIN:**

Q.19 (18 pnts): Convert the main program loop shown in the flow chart into valid C code. For the "Read the Inputs" block, you may assume that a function exists, `getInputs()`, which reads the inputs and saves the raw output of the reads to the global variables `uint8_t SS, PB`. **Your code must explicitly follow the flow chart.** Use the back of this page if more space is needed.

SS[1](P3.0)	PB[1](P3.1)	BLED[2](P2.0,P2.1)	Motor[3](P1.0,P1.1)
OFF	OFF/ON	RED	OFF
ON	OFF	GREEN	ON, Forward
ON	ON	GREEN	ON, Reverse

1: For Pushbutton: ON == Pressed, Slideswitch: ON == HIGH.
2: BLED RED/GREEN pin states are up to you.
3: If motor direction is 1, Motor will spin forward.

```
// Defined global variables
uint8_t SS, PB, timer_counter;
// Initialiations are here. You do not need to call them
while(1){
```