EE109: Introduction to Embedded Systems
Spring 2024 - Midterm Exam
03/26/24, 7PM – 8:40PM

[Complete all the information in the box below.]

Name:___**Solutions**_____

Student ID: _____

Email: _____**@usc.edu**

Lecture section (Circle One):

Redekopp	Redekopp	Weber	Puvvada
9:30 a.m.	11 a.m.	12:30 p.m.	2 p.m.

Ques.	Your score	Max score	Recommended Time
1		6	5 min.
2		4	5 min.
3		7	8 min.
4		10	15 min.
5		12	10 min.
6		10	12 min.
7		8	10 min.
8		5	5 min.
9		18	30 min.
Total		80	

All work MUST be on the FRONT (not back) of EXAM PAGES.

No Scratch work will be graded or viewed.

Do NOT write in the upper-right corner of the page with the QR code.

Do not write near this QR code.

1. (6 pts) Number Systems

1.1. Convert **203** decimal to an 8-bit unsigned binary number: __**1100 1011**_____

128 + 64 + 8 + 2 + 1

1.2. Convert **1001 1101 0100** binary to hexadecimal: 0x__**9 D 4**_____

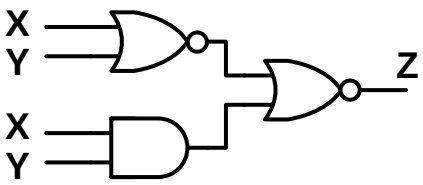
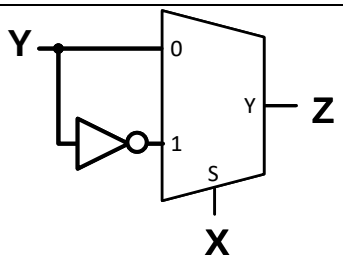
(

1.3. Using the **2's complement system** how many bits would be required for a number that could store any value in the range **-9 to +6**? __ **5** _____ bits

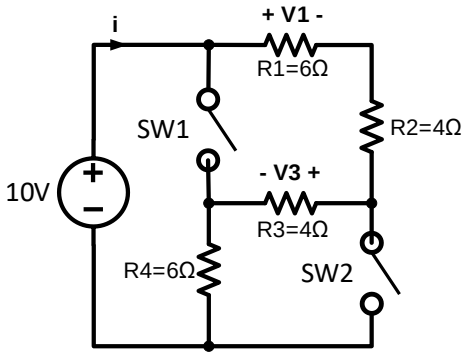
4-bits of 2's complement can represent the range -8 to +7 so we need 5 bits

2. (4 pts) Logic

For each circuit below, indicate True if it is equivalent to an XOR gate (i.e. $Z = X \oplus Y$).

	
Equivalent to $Z = X \oplus Y$: ☑ **T /** ❑ **F**	**Equivalent to $Z = X \oplus Y$:** ☑ **T /** ❑ **F**

3. (7 pts) Resistive Circuits



3.1. When both **SW1** and **SW2** are **OPEN** (disconnected) i is ___**0.5**_____ Amps.

When SW1 and 2 are open, R1, R2, R3, R4 are in series.

Thus V = 10 / R = 20 = 0.5A

3.2. When both **SW1** and **SW2** are **OPEN** (disconnected) what is V3: __10V * (4 / 20) = 2V_____

Because all resistors are in series when the switches are open, we can use the voltage divider equation: V3 = 10V * (4 / 20) = 2V

3.3. When both **SW1** and **SW2** are **CLOSED, R3 and R4** are in parallel. ☑ **T /** ❑ **F**

3.4. When both **SW1** and **SW2** are **CLOSED, R3 is in series with R1 and R2.** ❑ **T /** ☑ **F**

3.5. Consider the polarity of V3 shown in the diagram. When both **SW1** and **SW2** are **CLOSED** (connected), V3 will be _____. ❑ **positive** ☑ **negative**

Do not write near this QR code.

4. (10 pts.) Boolean Algebra: Use theorems to simplify the given equation for G to its minimal **SOP** form first, then in one step **ALSO show** the minimal **POS** expression (both POS and SOP). To get started follow the instructions in the 2nd column. For example, to start you need to use DeMorgan's theorem (potentially more than once) on the 2nd term of the equation. Then proceed to find the simplest SOP, then POS form. Show what theorems you are applying at each step (though you can apply 2 or 3 theorems per step). Write neatly. We **strongly** recommend you (PLEASE!!) plan your work on the scratch paper first to determine your approach but your **final solution must be** on this page. Use only the rows needed.

Step	Theorem(s) or Manipulation(s) Used
$G = A(B + \overline{C}D + \overline{A}B)(\overline{B} + \overline{C}D) + \overline{\left(D + (E \cdot \overline{[B + E]})\right)}$	
$G = A(B + \overline{C}D + \overline{A}B)(\overline{B} + \overline{C}D) + \_\_(\overline{D} \cdot (\overline{E} + [B + E]))\_$ Term 1 Term2	**DeMorgan's Theorem**
$G = A(B + \overline{C}D)(\overline{B} + \overline{C}D) + (\overline{D} \cdot (\overline{E} + [B + E]))$	**T9 applied somewhere in the first term**
$G = A(B \cdot \overline{B} + \overline{C}D) + (\overline{D} \cdot (\overline{E} + [B + E]))$	**You must apply T8' (not T10') to the first term**
$G = A\overline{C}D + (\overline{D} \cdot (\overline{E} + [B + E]))$	
$G = A\overline{C}D + (\overline{D} \cdot (1))$	
$G = A\overline{C}D + \overline{D} = (A + \overline{D})(\overline{C} + \overline{D})(D + \overline{D})$	
$G = (A + \overline{D})(\overline{C} + \overline{D})$	
$G =$____ $A\overline{C} + \overline{D}$ _____	**Minimal SOP .. and ..**
$G =$_____ $(A + \overline{D})(\overline{C} + \overline{D})$ _____	**Minimal POS**

Single-Variable Theorems

(T1)	X + 0 = X	(T1')	X • 1 = X	(Identities)
(T2)	X + 1 = 1	(T2')	X • 0 = 0	(Null elements)
(T3)	X + X = X	(T3')	X • X = X	(Idempotency)
(T4)	(X')' = X			(Involution)
(T5)	X + X' = 1	(T5')	X • X' = 0	(Complement)

Two- and Three-Variable Theorems

(T6)	X +Y = Y + X	(T6')	X • Y = Y • X	(Commutativity)
(T7)	(X+Y)+Z = X+(Y+Z)	(T7')	(X•Y) •Z = X• (Y•Z)	(Associativity)
(T8)	X•(Y+Z) = X•Y + X•Z	(T8')	X+(Y•Z) = (X+Y) • (X+Z)	(Distributivity)
(T9)	X + X•Y = X	(T9')	X • (X + Y) = X	(Covering)
(T10)	X•Y + X•Y' = X	(T10')	(X+Y) • (X+Y') = X	(Combining)
(T11)	X•Y+X'•Z+Y•Z = X•Y+X'Z	(T11')	(X+Y)•(X'+Z)•(Y+Z) = (X+Y)•(X'+Z)	(Consensus)

DeMorgan's Theorem

(X • Y)' = X' + Y'	(X +Y)' = X' • Y'	(DeMorgan's)

Do not write near this QR code.

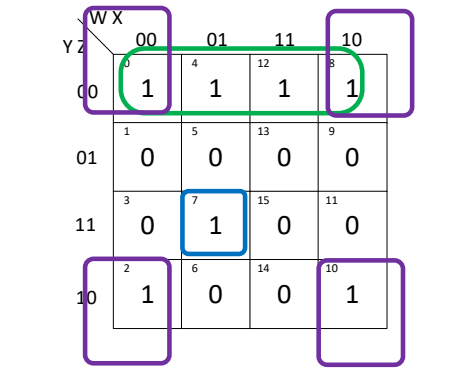
5. (12 pts.) Logic Simplification

Tricia attempted to find a **minimal SOP equation** for a function, **H**. Her equation below **correctly** expresses, H, but may **NOT** be minimal.

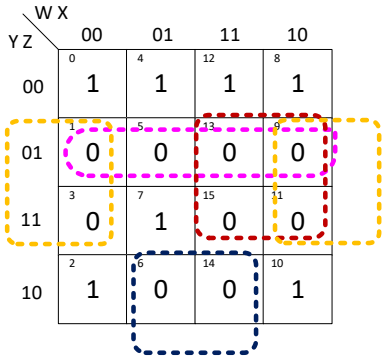
$$H(w,x,y,z) = w\,\overline{x}\,\overline{z} + \overline{x}\,y\,\overline{z} + \overline{y}\,\overline{z} + \overline{w}xyz$$

You MUST use the equation above to **construct and use the Karnaugh map below for the function, H**. Then **group** and **translate** to find or verify the minimal, **SOP** equation yielded by your Karnaugh Map and show your answer in the blank below to see if it agrees with the equation Tricia found. The truth table is optional. You may fill it out if it helps you organize and setup the K-Map.

W	X	Y	Z	H
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



Correct SOP: H = (Y'*Z')+(X'*Z')+(W'*X*Y*Z)



Correct POS for partial credit
H = (W'+Z')(X+Z')(Y+Z')(X'+Y'+Z)

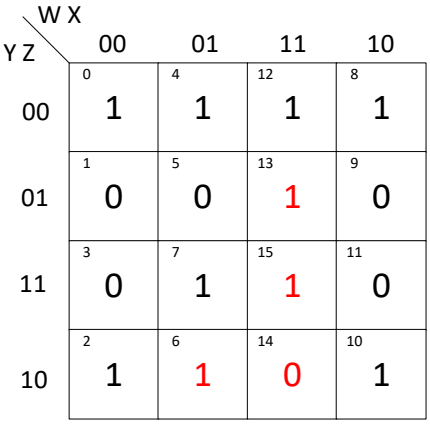
5.1. What is the minimal SOP equation you found for H.

H = _(Y'*Z') +(X'*Z')+(W'*X*Y*Z) _____

5.2. Now create a different function, G, by adding a new term to the original function, H, where ⊕ means XOR:

$$\boldsymbol{G} = \boldsymbol{H} + (wx \oplus y\overline{z})$$

(i.e.a new function G is created from Tricia's original equation, H, with a new term added (ORed) to it). Construct the K-Map for this new function, G. You need not group or translate. Simply, label the axes and fill in the 1s and 0s.



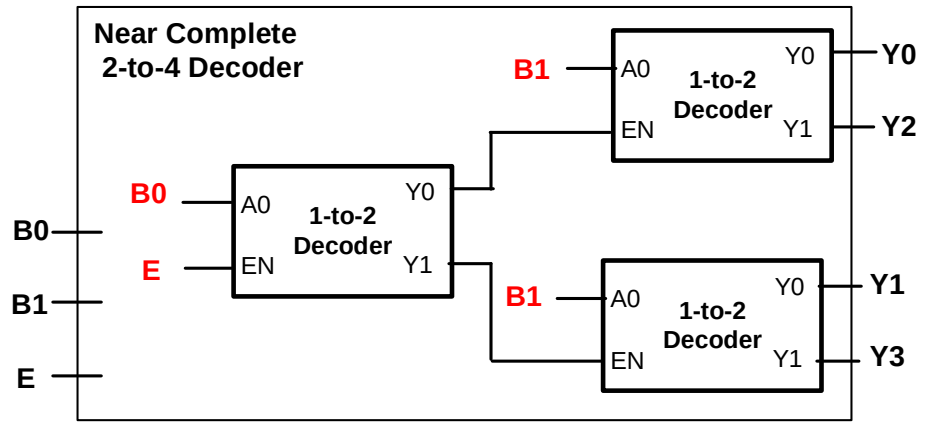
Do not write near this QR code.

6. **Decoders (10 pts)**. Design a 2-to-4 decoder with an enable by using 1-to-2 decoders with enables. Your goal is to decode the 2 bit number: **B1,B0** (where B1 is the MSB) and an enable **E** to the 4 outputs: **Y0, Y1, Y2, Y3**.

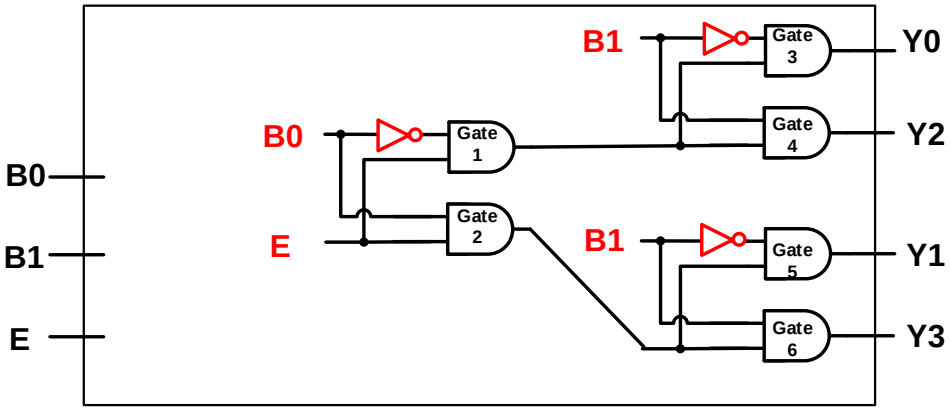
6.1. Complete the connections by labelling the blank inputs to the 1-to-2 decoder blocks with the appropriate inputs (e.g. E, B1, B0). **Note the ordering of the outputs in the diagram below**.

Truth Table of Desired 2-to-4 decoder

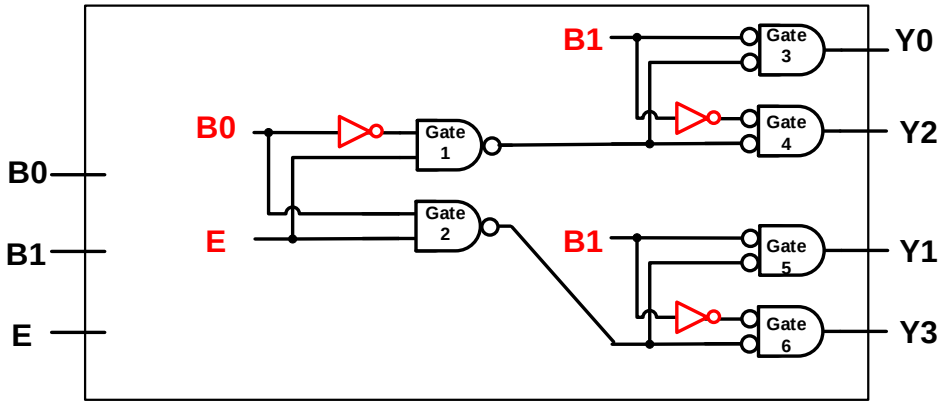
E	B1	B0	Y0	Y1	Y2	Y3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



6.2. Repeat the same design at the gate-level. The start of the internal gate design for each 1-to-2 decoder is shown below but each one is missing an inverter. Draw in **3 inverters** and complete the necessary connections (by labelling or drawing connections to the unconnected inputs) to form the desired 2-to-4 decoder. If multiple methods exist, choose any valid solution.



6.3. Tina Trojan changed Gate 1-6 as shown below. Again, by adding **3 inverters** and making appropriate connections in the design below, produce a circuit that matches the desired 2-to-4 decoder. If multiple methods exist, choose any valid solution.



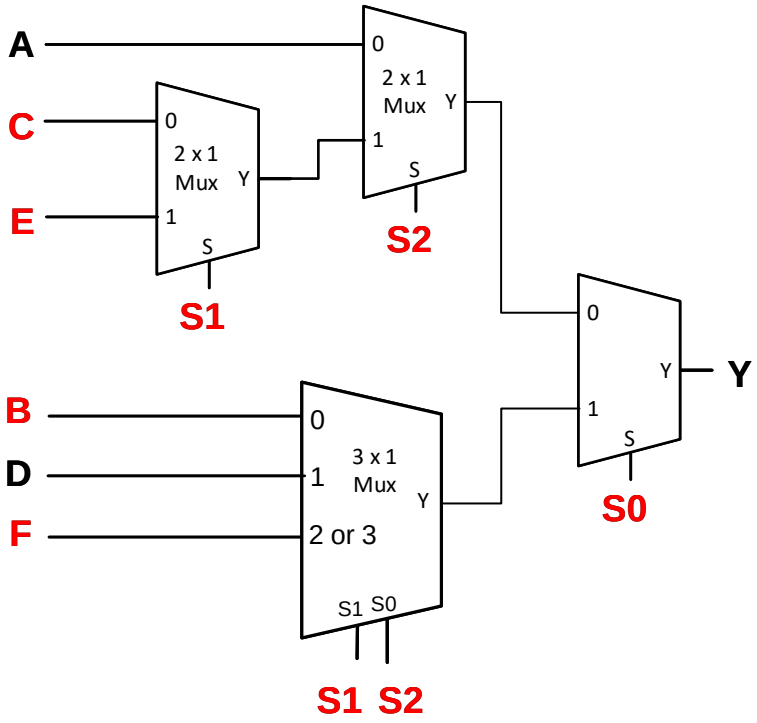
6.4. In the diagram for 6.3, gates **3-6** are what kind of gates. ❑ **NAND** / ☑ **NOR** / ❑ **neither**

Do not write near this QR code.

7. Muxes (8 pts)

Jean wanted to build a 6-to-1 mux design that follows the behavior described in the table below. She had several 2-to-1 muxes and **one 3-to-1 mux.** Fill in the shaded boxes (write darkly with your final answer so we can see your answer) with the data inputs **A, B, C, D, E,** or **F** and the select bits: **S2, S1, S0** to produce a design that matches the functionality described in the table. Note: the 3-to-1 mux will pass the input labelled "*2 or 3*" when the select number is either 2 or 3.

S2	S1	S0	Y
0	0	0	A
0	0	1	B
1	0	0	C
1	0	1	D
1	1	0	E
1	1	1	F



8. Lab Skills (5 pts)

8.1. The Digital Multimeter is best used to measure _____ voltage signals:

❑ **rapidly changing (Period < 0.1 seconds)** ☑ **roughly constant (Period > 10 sec)**

8.2. **Digital** signals generated by the Arduino are either 0V (for Logic 0) or ___ (for Logic 1):

❑ **5 mV** ❑ **100 mV** ❑ **1 V** ❑ **2 V** ☑ **5 V** ❑ **50 V**

8.3. The horizontal scale of an oscilloscope has units of:

❑ **voltage** ☑ **time** ❑ **current** ❑ **resistance**

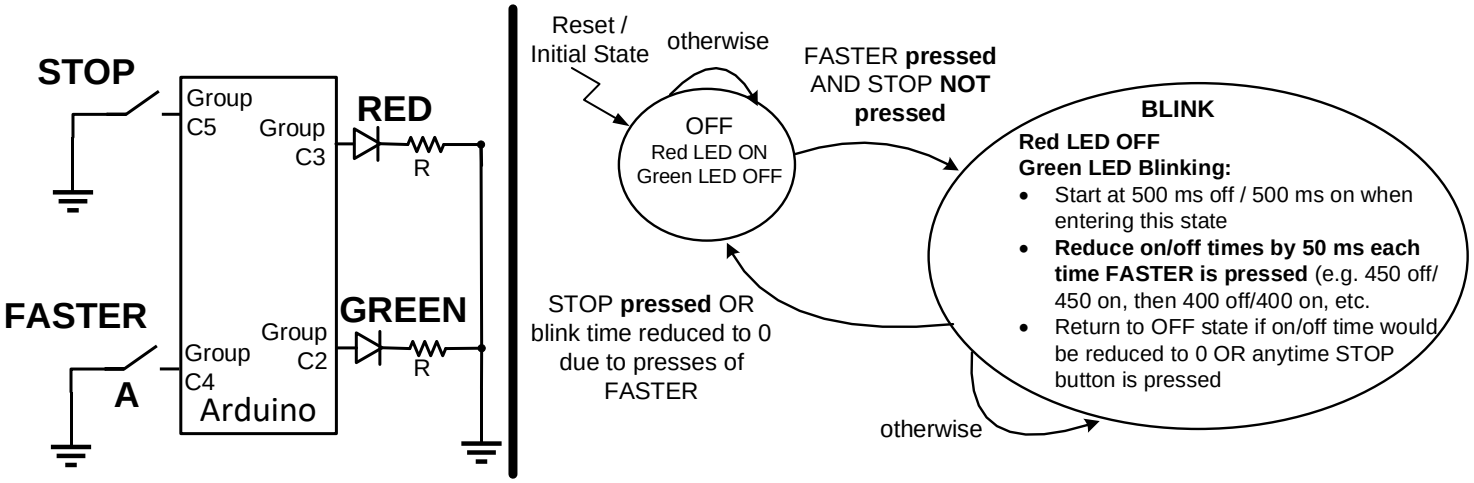
8.4. Single-acquisition/single-triggering on an oscilloscope is best used for capturing:

❑ **periodic signals** ☑ **aperiodic signals** ❑ **signals not measured relative to ground**

8.5. Prescalars for ADC and Timers are used to make the clock **SLOWER**. ☑ True ❑ False

Do not write near this QR code.

9. **(18 pts.) READ THE ENTIRE PAGE before solving.** We want to build a system with 2 states OFF and BLINK that makes an LED blink at faster rates every time we press a button. In the OFF state a Red LED (Group C, bit 3) should be ON and the Green (blinking) LED (Group C, bit 2) should be off. Pressing the FASTER button (Group C, bit 4) should cause a transition to the BLINK state where the Red LED should be off and the green LED should start by blinking at 1 HZ (500ms off, 500ms on). Each subsequent press of the FASTER button should **reduce the off and on blinking time by 50 ms** returning to the OFF state when the on/off time would be 0ms. Pressing the STOP button (Group C, bit 2) when in the BLINK state should cause the system to return to the OFF state and turn off the green LED.



We will use one of the Arduino hardware timers (similar to that used in your labs) to generate an interrupt EVERY 50 ms. Assume, a working function initialize_timer1() is provided to you to perform all necessary setup (i.e. you need not worry about configuring the timer's registers to generate the interrupt). **Complete** the Arduino C-code program on the following page that implements the behavior described above.

Note that this design may require debouncing, so in the code on the next page we had a `` `debounce(char bit)` `` function that should debounce and wait through the press of whichever input bit of Group C is specified by the argument, `bit`. You must decide when it is appropriate to call this function and what argument to pass it. **DO NOT use this function when it is NOT necessary for the operation of the system (or we will deduct points)**.

Important Requirements and Assumptions

- You must use the code shown and cannot alter its structure. Only fill in the blanks shown.
- Assume ALL necessary #includes are provided (but not shown in the code on the next page).
- You may use PC2, PC3, PC4, and PC5 constants, if desired.
- You may **NOT add other DELAY** (e.g. _delay_ms()) statements than the ones provided.

Complete your code on the page below!

Do not write near this QR code.

```
5   enum {OFF, BLINK};                // Assume appropriate #includes were provided on lines 1-4
6   volatile unsigned char count;
7   unsigned char state, max;
8
9   void init_timer(){ /* assume correct implementation for 50 ms interrupt interval */ }
10
11  void debounce(uint8_t bit){
12    _delay_ms(5); while(__(PINC & (1 << bit)) == 0______) {}   _delay_ms(5);
13  }
14  void transitionToOff() {        // common code when transitioning to the OFF state
15    state = OFF;
16    PORTC |= (1 << PC3);          // Turn Red ON
17    PORTC &= __~(1 << 2)___;      // Turn Green OFF
18    max = ___10____;                // update max appropriately
19  }
20  int main() {
21    state = OFF; count = 0; max = ___10______; // init max to correspond to 1 Hz blink rate
22    // Appropriate initialization for group C inputs
23    _PORTC |= (1 << PC4) | (1 << PC5)_____________________;
24    DDRC |=  (1 << PC2) | (1 << PC3);       PORTC |= (1 << PC3);  // Turn on red LED
25    init_timer();         // sets up timer to interrupt every 50ms
26    _sei()_______;      // enable global interrupts
27    while(1) {
28      char sample = _ PINC____;             // Sample buttons
29      if( state == OFF ) {
30        if ( (sample & (0x_30___)) == 0x_20_) {   // check if only FASTER button is pressed
31            _debounce(4)_________; // call debounce ONLY if necessary, leave blank otherwise
32            state = BLINK;
33            _PORTC &= ~(1 << PC3)_______;     // fill in the appropriate action
34      } }  // we put both braces on the same line to save space
35      else {                                  // in the BLINK state
36        if ( (sample & (1 << PC4)) == 0) {   // check if start of FASTER button press
37            _ debounce(4)______; // call debounce ONLY if necessary, leave blank otherwise
38            _max--___;         // necessary action to change blink rate
39            if(max == __0_______) { transitionToOff(); }
40        } else if ((sample & (1 << PC5)) == 0) {  // if STOP is pressed
41            _<blank>____; // call debounce ONLY if necessary, leave blank otherwise
42            transitionToOff();
43    } } } // we put the braces on the same line to save space
44    return 0;
45  } // end main
46  ISR(TIMER1_COMPA_vect) {
47    __count++_______;                   // Implement the necessary action
48    if(state == BLINK && count >= max) {
49      _count_____________ = 0;                   // Fill in the necessary variable
50      PORTC __ ^= (1 << PC2)__;             // Flip the Green LED
51  } } // we put the braces on the same line to save space
```