MULTIPLE-CHOICE ANSWERS

QT1 (1 pt) : **B. or “4”**

Of the **40** total segments available to us, only **36** are available (internal charge pump disables segments **0-3**) and, in static MUX mode, we associate **1** segment per **1** pin; therefore, given that we need **8** segments per **1** seven-segment digit, we are able to represent **4** full digits (integer division, “half” digits not applicable here).

QT2 (1 pt) : **D. or “4”**

Using the same logic as *QT1*, substituting **4** segments per **1** pin due to the 4-MUX mode, we are able to represent **18** possible digits, whereas in 3-MUX mode or **C.** would only allow for **13** full (no partial) digits.

The purpose of this exercise is to show that the total addressable segments increases with the higher MUX modes, thus demonstrating that, even though we will only utilize **7** digits for this exercise, there’s actually more possible segments to utilize.

QT3 (1 pt) : **B. or “128”**

While our LCD frequency of **240Hz** for a FPS of **30Hz** *technically* needs to be an approximate **133.33** clock divider value, the most approximate preset value for this is **128** or an actual LCD frequency of **250Hz**, which still yields us our minimum necessity of **240Hz** and is otherwise fine because we get a slightly greater FPS, ensuring that we avoid the flickering effect associated with LCD frame updates at *lower* than **30Hz**.

QT4 (1 pt) : **D. or “4, 3, 2, 1”**

Referring to the “NOTE” in the question’s prompt -”**2** segment pins per digit in 4-MUX”- we evidently need **14** segment pins to express **7** digits; HOWEVER, we have the complicating factor, as described in the associated text, that we are NOT allowed to use segment pins **0-3** due to the usage of the internal charge pump. Thus, we must offset our actual pin numbers by **3**, and we need pins **4-17** in order to address **7** digits. Given that pins are allocated in groups of **fours**, we will need to set the flags **LCDS4, LCDS8, LCDS12, LCDS16** or, numerically and respectively, flags **4, 3, 2, 1**.

QT5 (1 pt) : **D. or “0x1E”**

This is simply representing flags **LCDS4, LCDS8, LCDS12, LCDS16** in the *LCDAPCTL0/1* register scheme as its binary code. See student handout for register scheme.

QT6+7 (1 pt) :

Due to the exercise being student-driven, they have the option to select any number between **1-9** and are required to encode it as a series of segments and its associated bits. For the expediency of the instructor, a table for each digit and its segment-/binary-code is given here:

1	B | C	x0000-0110b or 0x06h
2	A | B | D | E | G	x0110-1011b or 0x6Bh
3	A | B | C | D | G	x0010-1111b or 0x2Fh
4	B | C | F | G	x0011-0110b or 0x36h
5	A | C | D | F | G	x0011-1101b or 0x3Dh
6	A | C | D | E | F | G	x0111-1101b or 0x7Dh
7	A | B | C	x0000-0111b or 0x07h
8	A | B | C | D | E | F | G	x0111-1111b or 0x7Fh
9	A | B | C | D | F | G	x0011-1111b or 0x3Fh

Common issues that could result in students not seeing their character displayed properly:

- Ensure that MUX mode and segment pins are set correctly (see *QT2* and *QT5*)
- Ensure that LCDSON and LCDON were, in fact, turned on/set.
- Ensure that student used Boolean OR operators in order to assign values to the LCDACTL and NOT just a regular assignment (bis.b)
- Ensure that Boolean OR is used to construct the bit patterns and NOT a logical OR (**||**)

SHORT RESPONSE ANSWERS

(QC1 on abridged) (4 pts) :

+1 pt for restating, in some form, what a “flag” is or associating it as bits or codes that do something in a controller, system, or object.

+1 pt for restating what “memory-mapped hardware” is, in that it’s a space in memory to control hardware, serving as a “bridge” or “intermediary” between your “core” microprocessor components and the richer features of a microcontroller.

+1 pt for associating that the memory registers for hardware are composed of flags that act as the parameters for operational/conditional performance of said hardware

+1 pt for effort or an attempt, awarded for both wrong and right answers.

QC1 (QC2 on abridged) (3 pts) :

The answer here is self-evident, given that “MUX” is a shorthand for “multiplex;” further evidence is given in the prompt’s hint to “find the segment pins -Sx- on data sheet,” which would evidently show that the segment pins share the same pins as port peripherals. Student should demonstrate comprehension that pins on the MSP430 are multifunctional and that accessing these pins to drive a display requires selecting the LCD function explicitly, so multiplexing “frees up” pins.

Otherwise, points should also be awarded for utilizing the answers from *QT1* and *QT2* to acknowledge that multiplexing segment pins allows us to address more possible segments (in conjunction with the COM pins, though that admission is excluded for sake of brevity). Additionally, this answer is explicitly given in the *LCDAPCTL0/1* section.

+2 pts for either of the above or equivalent answers; the additional **+1 pt** is purely effort-based and awarded even in the event of egregiously wrong answers, as a failure to understand need not admonish one’s efforts otherwise.

QC2 (QC3 on abridged) (2 pts) :

This is more a short-response demonstrating effort than correctness, meant to be purposefully vague or otherwise not explicitly described in the lab text, in order to provide the student an opportunity to “reverse engineer” solutions by taking a design choice and analyzing what possible causes and problems could serve as the impetus for this decision. Accordingly, award points liberally as measures (and acknowledgements) of effort.

Accepted possible answers include:

- Animation, allows for flashing text.
- Mutability, allows for memory registers associated with display to be preserved but still allows for its segments to be displayed as “cleared” or “off.”
- Low-power modes, serving a similar application to “suspended” states for CPUs, allowing the device to still be “on” and “ready” without necessarily being too active.
- Configuration, a slightly more technical answer but the manner in which the LCD_A handles itself is particularly convoluted: display memory registers are left in an *indeterministic* state upon powering up and, due to aspects like the segment pin group offset (as explained in *QT4*), you have segments enabled and have their associated pins turned on despite not necessarily wanting them to be. Therefore, having the controller on but not the displaying allows you to “handle” these peculiarities before they are visibly caught by an end-user.

*QC3 (**full version only**) (2 pts)* :

In a TN-LCD (assuming no backlight), a segment is displayed when a voltage bias applied across the electrode backplanes straightens the helical structure of the liquid crystal’s molecules, thereby NOT altering the orientation of incident light traveling through it and, as a result, NOT having it at the correct angle to pass through the second polarized lens in order to

be reflected back off from the mirror on the opposite side. Therefore, that segment appears dark due to light being “trapped,” which is perceived as a “lit” or “on” segment.

The question prompt asks for what description of what happens when a segment is NOT displayed. Evidently, the answer is the converse of the above description: the lack of a bias allows the liquid crystal to return to its twisted structure, which reorientates incident light passing through it to be in at an angle such that it can pass through the second polarized lens, hitting the mirror behind it and bouncing back and out the LCD front glass. Therefore, the segment appears NOT lit when incoming light travels into and back out of the LCD.

+1 pt for the correct answer; **+1 pt** for an effort made.

*QC4 (**full version only**) (2 pts)* :

Speaking specifically on non-backlit TN-LCDs, their energy-efficiency is due to, at its simplest, the fact that its “illuminating” element is all ambient lighting and that the voltage bias to establish whether light is allowed to pass through the LCD can be achieved with an incredibly small potential difference, which can be done through very little current requirements.

Otherwise, if they are comparing it to the relative inefficiency of a CRT, they could explain how CRTs are thermally-operative devices (the heat produced by a tungsten coil is responsible for the emission of free electrons from the cathode) and how this energy mechanism requires higher wattage than the simple electrical modulation of LCDs (think of comparing the power requirements of high-pressure sodium or fluorescent lights to that of modern LEDs).

+1 pt for the correct answer; **+1 pt** for an effort made.

EXERCISE (1) + (2) CODE:

```
#include <msp430.h>

#define         SW1             (P1IN&BIT0)                     // Macro for SW1 on MSP430

int main(void)
{
        WDTCTL = WDTPW | WDTHOLD;

        // Ensure that student is using the library macro for LCDM3 explicitly
        unsigned char* mem = (unsigned char*)(&LCDM3);

        int i;
        for ( i=0; i<20; i++ )
                LCDMEM[i] = 0x00;                               // Clears junk data in memory

        LCDACTL |= LCD4MUX;
        LCDACTL |= LCDFREQ_128;

        P5SEL |= (0x10 | 0x08 | 0x04);                          // ESSENTIAL : if missing,
                                                                //    behavior is undetermined
        LCDAPCTL0 = 0x1E;

        LCDACTL |= (LCDSON | LCDON);

        mem[0] = (0x01 | 0x02 | 0x04 | 0x08 | 0x40 | 0x10);     // Should print “0” in 1st digit

        P1DIR &= ~BIT0;
        while(1)
        {
                If ( SW1 == 0 )
                {
                        for ( i=2000; i>0; i– );

                        // Prints “F” in 2nd digit when SW1 is depressed (remains thereafter)
                        if ( SW1 == 0 )
                                Mem[1] = (0x01 | 0x40 | 0x20 | 0x10);
                }
        }
}
```