Note: student answered Problem 5 by writing the code in pencil on the paper exam, but it's easier to show the solution typed.

Problem 5:

```
#include <p18f4520.h>
#include <delays.h>
// #pragma stuff not shown

void main(void) {
    // Use the default oscillator speed as done in Lab 3 (which btw is 1MHz).
    // In lab 3 it said Delay10KTCYx(50); was 2 seconds, so
    //           10,000*50 = 500,000 instructions = 2 seconds
    // Therefore…
    //      3 seconds = 750,000 instructions
    //    1.5 seconds = 375,000 instructions

    ADCON1 = 0b00001111; // Sets all the pins to digital
    TRISC = 0b00000000; // Sets all the digital PORTC pins to outputs

    while (1) {
        //State 1 - Only LEDs RC0 and RC2 are on for 1.5 seconds
        PORTC = 0b0101;
        Delay10KTCYx(37);  // Delay for 1.5 seconds (25 = 1, 12 = 0.5)
        // or Delay10KTCYx(38); is also correct
        // Note: Technically it is not allowed to use floats:
        // Delay10KTCYx(37.5); be we didn't count that wrong
        // So to be "perfectly" correct you should actually use:
        // Delay10KTCYx(37); then add this...
        Delay1KTCYx(5);
/* To be fair to students we counted Delay10KTCYx(37);  Delay10KTCYx(38); or
Delay10KTCYx(37.5); correct without needing the Delay1KTCYx(5); command since
in Lab 3 it was only a simple integer change to the argument and students
didn't know the details about using delays yet before Exam 1.  In the future
though know that in delay functions the argument must be an integer number
(unsigned char) between 1 and 255. */

        //State 2 - Only LEDs RC1 and RC3 are on for 3 seconds
        PORTC = 0b1010;    //
        Delay10KTCYx(75);  // Delay for 3 seconds (50 = 2, 25 = 1)
    }
}
```

Note that with code there are MANY solutions. Here is one solution:

Problem 6a:

```
#include <stdio.h>
char primes[] = {2, 3, 5, 7, 11, 13};
void printPrimes(char value);

void main(void) {
    char value;
    for (value = 1; value <= 24; value++) {
        printf("\nThe prime factors of %d are: ", value);
        printPrimes(value);
    }
    while (1);
}

void printPrimes(char value) {
    char primeIndex;
    for (primeIndex = 0; primeIndex < sizeof(primes); primeIndex++) {
        char prime = primes[primeIndex];
        if (value % prime == 0) {
            printf("%d ", prime);
        }
    }
}
```

Problem 6b:

```c
#include <stdio.h>
char primes[] = {2, 3, 5, 7, 11, 13};
void printPrimes(char value);

void main(void) {
    char value;
    for (value = 1; value <= 24; value++) {
        printf("\nThe prime factors of %d are: ", value);
        printPrimes(value);
    }
    while (1);
}

void printPrimes(char value) {
    char primeIndex;
    for (primeIndex = 0; primeIndex < sizeof(primes); primeIndex++) {
        char prime = primes[primeIndex];
        if (value % prime == 0) {
            printf("%d ", prime);
            printPrimes(value / prime);
            break;
        }
    }
}
```

Yes, we are aware that 6b is a hard problem. It was worth fewer points and is intended to be a challenge. The solution above is one of **many** possible solutions. Neat that it only adds 2 lines of code. It calls the function again to get a fresh start and ends the current loop (note the `break;` statement could instead be a `return;` statement). The solution to 6b that almost all students used was to make a while loop where I have the yellow highlights above that looks for repeat primes, prints them, and then makes the value smaller as it does so until value % prime is not 0. Code that might look something like this:

```
void printPrimes(char value) {
    char primeIndex;
    for (primeIndex = 0; primeIndex < sizeof(primes); primeIndex++) {
        char prime = primes[primeIndex];
        while (value % prime == 0) {
            printf("%d ", prime);
            value = value / prime;
        }
    }
}
```

Output

```
The prime factors of 1 are:
The prime factors of 2 are: 2
The prime factors of 3 are: 3
The prime factors of 4 are: 2 2
The prime factors of 5 are: 5
The prime factors of 6 are: 2 3
The prime factors of 7 are: 7
The prime factors of 8 are: 2 2 2
The prime factors of 9 are: 3 3
The prime factors of 10 are: 2 5
The prime factors of 11 are: 11
The prime factors of 12 are: 2 2 3
The prime factors of 13 are: 13
The prime factors of 14 are: 2 7
The prime factors of 15 are: 3 5
The prime factors of 16 are: 2 2 2 2
The prime factors of 17 are:
The prime factors of 18 are: 2 3 3
The prime factors of 19 are:
The prime factors of 20 are: 2 2 5
The prime factors of 21 are: 3 7
The prime factors of 22 are: 2 11
The prime factors of 23 are:
The prime factors of 24 are: 2 2 2 3
```