Assignments 7 & 8

Create a menu driven program for each assignment. The assignments are described below.

Remember that procedures must accomplish one thing. This means that you may have to create procedures to support the procedures outlined below.

At all times, you will use INVOKE and PROTO for your procedures. The ONLY exception is the Irvine Library – you MUST pass all information to Irvine instructions in registers as outlined in Chapter 5.

Global Variables are strictly not allowed. If you declare something in .data above main, you can only pass it to your procedures, you CANNOT explicitly reference it in your procedures. You may use local variables and the stack as necessary. You may have a .data in your procedures.

Assignment 7

Part 1: FindPrimes (25pts)

1. Write a procedure (FindPrimes) that **determines** the prime numbers between the unsigned values of 2 to 1000 (inclusive).

    a. Use the Sieve of Eratosthenes to find all the prime numbers in this range. (lots of info online and pseudocode provided below)
    b. Display your results using the procedure developed in step 3 below.
    c. Simply coding an array with a bunch of primes in it will result in a zero grade for part 1 and 2 of this assignment.

2. Write a procedure (DisplayPrimes) that displays how many primes are in the range 2 to n (n<=1000) in the following format. **DO NOT USE DUMPMEM**. There should be 5 prime numbers per line except for the last line, which may or may not have 5 numbers. Ensure that your numbers line up in columns as shown below. A string of 5 spaces will NOT work.

    The user will enter n, there should be error checking to ensure the user does not enter an invalid number, i.e. negative or too large.

3. If you use an algorithm based on the square root of n, you may hard code the floor of square root of 1000.

For example. n = 25
The output shall look like the following:

```
There are 9 primes between 2 and n (n = 25)
------------------------------------------------
2     3     5     7     11
13    17    19
```

Assignment 8

Euclid (15pts)

Write a recursive implementation of Euclid's Algorithm for finding the greatest common divisor (GCD) of two integers. You can find pseudocode for this algorithm online (be sure to include in your program description where you found the code). This procedure will use the procedure written in Assignment 7. The output shall be of the following format

```
Number #1 Number #2 GCD GCD Prime?
-------------------------------------------------
6          15         3 Yes

Do you wish to enter another pair (Y/N) Y
4          3          1 No

Do you wish to enter another pair (Y/N) Y
8          16         8 No

Do you wish to enter another pair (Y/N) N
```

You may re-use your code from Assignment 7 for determining if a number is prime.

Part 3 – Matrix of words (10pts)

At all times in this assignment, you must use the 2D row major format.

Create a procedure that generates a 5 x 5 matrix of randomly chosen capital letters. Each letter will have a 50% chance of being a vowel. (HINT: mod 2 might be helpful here)

Once you have your 5 x 5 matrix is built, print it out. Use a procedure for this.

Then build a procedure that will go through your matrix and find sets of letters which are comprised of 5 letters, 2 of which will be vowels. The order of the letters in the set is not important. For the following example, ABOST and TSOBA are equivalent. If there are no words, then the output should indicate that no words were found. There will be a maximum of 12 sets of letters (5 rows, 5 columns, 2 diagonals)

EXAMPLE:

```
The matrix is:

A D T R G
B K L E B
O U I R Q
S T V X P
T Y S Z H

The words from this matrix is/are:
ABOST, AKIXH, GEITT
```

Specifications (For both assignments):

1. Use an appropriately sized array
2. Use row-major format, if required.
3. Only instructions discussed in class
4. As always commenting and style are important. Don't forget to include the headers for your procedures.
5. You must use INVOKE and PROTO unless calling an Irvine library instruction. **No use of USES.**
6. No . directives. i.e. .IF, etc s.
7. There is a lot of flexibility in how this assignment can be coded.
8. You may use any command we have discussed in class and the Irvine Library. Remember any procedure from the Irvine library uses the registers, not the stack. You must use call when using these procedures.
9. YOU MUST FULLY DOCUMENT ALL PROCEDURES. FAILURE TO DO SO WILL RESULT IN A GRADE REDUCTION.
10. NO SPAGHETTI CODE. Be concise and limit your procedures to accomplishing 1 basic thing.
11. Implement error checking for the menu and all user input.