Introduction

The purpose of this assignment is to reinforce concepts around procedures, nested loops, and data validation (CLO 3, 4). Please be sure to thoroughly check the rubric - there are some specific requirements with point penalties if they are not satisfied.

1. Designing and implementing procedures
2. Designing and implementing loops
3. Writing nested loops
4. Understanding data validation

What you must do

**Program Description**

Write a program to calculate prime numbers. First, the user is instructed to enter the number of primes to be displayed, and is prompted to enter an integer in the range [1 ... 200]. The user enters a number, *n*, and the program verifies that 1 ≤ *n* ≤ 200. If *n* is out of range, the user is re-prompted until they enter a value in the specified range. The program then calculates and displays the all of the prime numbers up to and including the *nth* prime. The results must be displayed 10 prime numbers per line, in ascending order, with at least 3 spaces between the numbers. The final row may contain fewer than 10 values.

**Program Requirements**

1. The programmer’s name and program title must appear in the output.
2. The **counting loop** (1 to *n*) must be implemented using the **LOOP** instruction.
3. The **main** procedure must consist of only procedure calls (with any necessary framing). It should be a readable "list" of what the program will do.
4. Each procedure will implement a section of the program logic, i.e., each procedure will specify how the logic of its section is implemented. The program must be modularized into *at least* the following procedures and sub-procedures:
   * **introduction**
   * **getUserData** - Obtain user input
     + **validate** - Validate user input ***n***is in specified bounds
   * **showPrimes** - display ***n*** prime numbers; utilize counting loop and the **LOOP** instruction to keep track of the number primes displayed; candidate primes are generated within counting loop and are passed to **isPrime** for evaluation
     + **isPrime** - receive candidate value, return boolean (0 or 1) indicating whether candidate value is prime (1) or not prime (0)
   * **farewell**

* The upper and lower bounds of user input must be defined as **constants**.
* The **USES** directive is not allowed on this project.
* If the user enters a number outside the range [1 ... 200] an error message must be displayed and the user must be prompted to re-enter the number of primes to be shown.
* The program **must** be fully documented and laid out according to the [CS271 Style Guide](https://canvas.oregonstate.edu/courses/1910943/files/94857574/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

. This includes a complete header block for identification, description, etc., a comment outline to explain each section of code, and proper procedure headers/documentation.

**Notes**

1. For this program, you may use global variables instead of passing parameters in memory or on the stack. This is a one-time relaxation of the standards so that you can get accustomed to using procedures.
2. A number *k* is prime if it is an integer with exactly 2 distinct divisors: 1 and *k*. Note that this implies that
   1. 1 is not prime.
   2. *k* must be positive.
3. There are several ways to make your **isPrime** procedure efficient. Feel free to discuss algorithms on the Ed Discussion board and Teams.
4. Check the [Course Syllabus](https://canvas.oregonstate.edu/courses/1910943/files/94871569/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

 for late submission guidelines.

1. Find the assembly language instruction syntax and help in the [CS271 Instructions Guide](https://canvas.oregonstate.edu/courses/1910943/files/95544921/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

.

1. To create, assemble, run,  and modify your program, follow the instructions on the course [Syllabus Page](https://canvas.oregonstate.edu/courses/1910943/assignments/syllabus)’s "Tools" tab.

Resources

Additional resources for this assignment

* [Project Shell with Template.asm](https://canvas.oregonstate.edu/courses/1910943/files/94857201/download?wrap=1)[Download Project Shell with Template.asm](https://canvas.oregonstate.edu/courses/1910943/files/94857201/download?download_frd=1)
* [CS271 Style Guide](https://canvas.oregonstate.edu/courses/1910943/files/94857574/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

* [CS271 Instructions Reference](https://canvas.oregonstate.edu/courses/1910943/files/95544921/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

* [CS271 Irvine Procedure Reference](https://canvas.oregonstate.edu/courses/1910943/files/94857677/download?wrap=1)

[Actions](https://canvas.oregonstate.edu/courses/1910943/assignments/8972780)

What to turn in

Turn in a single .asm file (the actual Assembly Language Program file, not the Visual Studio solution file).  File must be named "Proj4\_ONID.asm" where ONID is your ONID username. Failure to name files according to this convention may result in reduced scores (or ungraded work).

Example Execution

User input in this example is shown in ***boldface italics***.

Prime Numbers Programmed by Euclid  
  
Enter the number of prime numbers you would like to see.  
I’ll accept orders for up to 200 primes.  
  
Enter the number of primes to display [1 ... 200]: ***301***  
No primes for you! Number out of range. Try again.  
Enter the number of primes to display [1 ... 200]: ***0***  
No primes for you! Number out of range. Try again.  
Enter the number of primes to display [1 ... 200]: ***31***  
  
2 3 5 7 11 13 17 19 23 29  
31 37 41 43 47 53 59 61 67 71  
73 79 83 89 97 101 103 107 109 113  
127  
  
Results certified by Euclid. Goodbye.

Extra Credit (Original Project Definition must be Fulfilled)

To receive points for any extra credit options, you **must** add one print statement to your program output **per extra credit** which describes the extra credit you chose to work on. You ***will not receive extra credit points*** unless you do this. The statement must be formatted as follows...

--Program Intro--  
\*\*EC: DESCRIPTION  
  
--Program prompts, etc—

**Extra Credit Options**

1. Align the output columns (the first digit of each number on a row should match with the row above). (1pt)
2. Extend the range of primes to display up to 4000 primes, shown 20 *rows* of primes per page.  The user can “Press any key to continue …” to view the next page.  Since length of the numbers will increase, it’s OK to display fewer numbers per line. (2pt)

[8085 program to determine if the number is prime or not - GeeksforGeeks](https://www.geeksforgeeks.org/8085-program-to-determine-if-the-number-is-prime-or-not/#:~:text=Algorithm%3A%201%20Take%20n%20as%20input%202%20Run,2%2C%20then%20the%20number%20is%20prime%2C%20else%20composite)