Program #4

Due Feb 13 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** asm

Due: Week 6, Sunday, 11:59 PM Pacific USA Time Zone

Objectives

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

Problem Definition

Write a program to calculate composite numbers. First, the user is instructed to enter the number of composites to be displayed, and is prompted to enter an integer in the range [1 .. 300]. The user enters a number, **n**, and the program verifies that 1 ≤ **n** ≤ 300. 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 all of the composite numbers up to and including the **n**th composite. The results should be displayed 10 composites per line with at least 3 spaces between the numbers.

Requirements

1. The programmer's name must appear in the output.
2. The counting loop (1 to **n**) must be implemented using the MASM **loop** instruction.
3. The **main** procedure must consist (mostly) of procedure calls. 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
     - validate

    - showComposites
        - isComposite
    - goodbye
5. The upper limit should be defined and used as a constant.
6. Data validation is required. If the user enters a number outside the range [1 .. 300] an error message should be displayed and the user should be prompted to re-enter the number of composites.
7. Each procedure must have a procedure header that follows the format discussed during lecture.
8. The usual requirements regarding documentation, readability, user-friendliness, etc., apply.

Notes

1. For this program, you may use global variables instead of passing parameters. This is a one-time relaxation of the standards so that you can get accustomed to using procedures. In future homework, global variables will not be allowed.
2. A number **k** is composite if it can be factored into a product of smaller integers. Every integer greater than one is either prime or composite. Note that this implies that
    - 1 is not composite.
    - The number must be positive.
3. If you choose to use the LOCAL directive while working on this program be sure to read section 8.2.9 in the Irvine textbook. LOCAL variables will affect the contents of the system stack!

Example Program Operation

```
Welcome to the Composite Number Spreadsheet
Programmed by Author Name
This program is capable of generating a list of composite numbers.
Simply let me know how many you would like to see.
I'll accept orders for up to 300 composites.
How many composites do you want to view? [1 .. 300]: 400
Out of range. Please try again.
How many composites do you want to view? [1 .. 300]: 0
Out of range. Please try again.
How many composites do you want to view? [1 .. 300]: 23
4   6   8   9   10   12   14   15   16   18
20   21   22   24   25   26   27   28   30   32
33   34   35
Thanks for using my program!
```

Extra Credit Option (original definition must be fulfilled)

- (2 pts) When the program runs, give the user the option to display only composite numbers that are also odd numbers. The user should get a choice when the program first starts (e.g. enter 0 to view all composite numbers or enter 1 to view only odd composites). For example, if the user chooses to view only odd composites, they would see the numbers 9, 15, 21, 25, 27, 33, 35, 39, 45...

Remember, in order to ensure you receive credit for any extra credit work, 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--
```

Please refer back to the documentation for Program 1 to see a sample of the extra credit format.

Program 4 Rubric

Criteria	Ratings			Pts
Files Correctly Submitted Submitted file is correct assignment and is an individual .asm file.	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Program Assembles & Links Submitted program assembles and links without need for clarifying work for TA and/or messages to the student. This assumes the program is actually an attempt at the assignment. Non-attempts which compile/link earn no points.	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Documentation - Identification Block - Header Name, Date, Program number, etc as per syllabus are included in Identification Block	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Documentation - Identification Block - Program Description Description of functionality and purpose of program is included in identification block.	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Documentation - Procedure Headers Procedure headers describe functionality and implementation of program flow. Should also list pre- and post-conditions and registers changed.	**4 pts** **Full Marks**	**2 pts** **Headers without Conditions** Descriptive headers but lacking pre- and post-conditions and 'registers changed'	**0 pts** **No Marks**	4 pts

Criteria	Ratings		Pts
Documentation - In-line Comments In-line comments contribute to understanding of program flow (from section comments) but are not line-by-line descriptions of moving memory to registers.	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Verification - Program Executes Program executes and makes some attempt at the assigned functionality.	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
Completeness - Displays Title & Programmer Name Program prints out the programmer's name and Program Title	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Completeness - Displays Introduction Displays program introduction. Program introduction should describe functionality of ▶ ram.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Completeness - Prompts for Input (with Bounds) Prompts user to enter data, specifying bounds of acceptable inputs.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts

Criteria	Ratings			Pts
Completeness - Gets data from User Utilizes ReadInt or ReadDec to receive user input. Saves values in appropriately-named identifiers for validation.	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Completeness - Validates User Data Validates that user-entered values are within the advertised limits.	**2 pts** **Full Marks**	**1 pts** **Partial validation** Neglects to check edge cases.	**0 pts** **No Marks**	2 pts
Completeness - Displays Results	**2 pts** **Full Marks**		**0 pts** **No Marks**	2 pts
Completeness - Displays closing message	**1 pts** **Full Marks**		**0 pts** **No Marks**	1 pts
Correctness - Displays correct number of composite numbers	**4 pts** **Full Marks**	**3 pts** **Mostly Correct** Edge cases are neglected or display incorrectly.	**0 pts** **No Marks**	4 pts
Correctness - Calculations are Correct All displayed values are in fact composites, beginning with the number '4' (if running in extra credit mode, the numbers should start with '9')	**6 pts** **Full Marks**		**0 pts** **No Marks**	6 pts

Criteria	Ratings				Pts
Correctness - Numbers are displayed 10 per line	**2 pts** **Full Marks**		**0 pts** **No Marks**		2 pts
Correctness - Partial lines displayed correctly	**1 pts** **Full Marks**		**0 pts** **No Marks**		1 pts
Requirements - Limit is a CONSTANT Max number allowable is defined and used as a CONSTANT.	**1 pts** **Full Marks**		**0 pts** **No Marks**		1 pts
Requirements - Program is well-modularized Program is divided into logical sections based on function. Program implements all required procedures with good logical flow according to class lectures. Note: as a reminder, the required procedures are introduction, getUserData, validate, showComposites, isComposite, goodbye. Adding extra procedure is allowed. ▶	**10 pts** **Full Marks**	**8 pts** **Lacking some Procedures** Well modularized, but lacking all required procedures	**2 pts** **No procedures** Well modularized, but fails to utilize procedures	**0 pts** **No Marks**	10 pts
Requirements - Counted loop uses LOOP instruction when displaying the correct number of composite numbers	**1 pts** **Full Marks**		**0 pts** **No Marks**		1 pts

Criteria	Ratings		Pts
Coding Style - Uses appropriately named identifiers Identifiers named so that a person reading the code can intuit the purpose of a variable, constant, or label just by reading its name.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Coding Style - Readability Program uses readable white-space, indentation, and spacing as per the Indentation Style Guide. Logical sections are separated by white space.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Output Style - Readability Program output is easy to read	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Extra Credit Program has an extra mode that will only display composite numbers that are also odd numbers (+2). For example, if the user chooses to operate in extra credit mode and asks to view 3 composite numbers, they will see "9, 15, 21".	**0 pts** **Full Marks**	**0 pts** **No Marks**	0 pts
Late Penalty Remove points here for late assignments. (Enter negative point value)	**0 pts** **Full Marks**	**0 pts** **No Marks**	0 pts
			Total Points: 50