Program #2

Due Jan 23 by 11:59pm **Points** 50 **Submitting** a file upload **File Types** asm

Due: Week 3, Sunday 11:59 PM Pacific USA time zone

Objectives

1. Getting string input
2. Designing and implementing a counted loop
3. Designing and implementing a post-test loop
4. Keeping track of a previous value
5. Implementing data validation

Problem Definition

Write a program to calculate Fibonacci numbers.

- Display the program title and programmer's name. Then prompt the user for their name and greet them (by name).
- Prompt the user to enter the number of Fibonacci terms to be displayed. Advise the user to enter an integer in the range [1 .. 46].
- Get and validate the user input (n).
- Calculate and display all of the Fibonacci numbers up to and including the nth term. The results should be displayed 4 terms per line with at least 5 spaces between terms.
- Display a parting message that includes the user's name, and terminate the program.

Requirements

1. The programmer's name and the user's name must appear in the output.
2. The loop that implements data validation must be implemented as a post-test loop.
3. The loop that calculates the Fibonacci terms must be implemented using the MASM **loop** instruction.
4. Numeric user input must be aquired using the ReadInt Irvine procedure (as a signed integer).
5. The **main** procedure must be modularized into at least the following sections (procedures are not required in this program):

- introduction
- displayInstructions
- getUserInfo
- displayFibs
- goodbye

6. Note that each of the above sections (introduction, displayInstructions, getUserInfo, etc) needs to have a header block explaining its purpose.
7. Recursive solutions are not acceptable for this assignment. This one is about iteration.
8. The upper limit must be defined and used as a constant.
9. The usual requirements regarding documentation, readability, user-friendliness, etc., apply.

Notes

1. It is not necessary to store the Fibonacci numbers in an array. The terms may be displayed as they are generated.
2. The second-order Fibonacci sequence is defined as:
    1. The first two terms are both 1.
    2. All other terms are calculated as the sum of the two previous terms.
3. The reason for restricting n to [1 .. 46] is that the 47th Fibonacci number is too big for DWORD data type.

Example Program Operation

```
nacci Numbers
rammed by Leonardo Pisano
What’s your name? Paul
Hello, Paul
Enter the number of Fibonacci terms to be displayed.
Provide the number as an integer in the range [1 .. 46].
How many Fibonacci terms do you want? 50
Out of range. Enter a number in [1 .. 46]
How many Fibonacci terms do you want? -6
Out of range. Enter a number in [1 .. 46]
How many Fibonacci terms do you want? 14
1     1     2     3
5     8     13     21
34     55     89     144
233     377
Results certified by Leonardo Pisano.
Goodbye, Paul.
```

Extra Credit Option (original definition must be fulfilled)

- (1 pt) Display the numbers in aligned columns.

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 2 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.	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 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.	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Documentation - Section Comments Code section headers describe functionality and implementation of program flow. Should mirror the style guide image.	**4 pts** **Full Marks**	**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.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Verification - Program Executes Program executes and makes some attempt at the assigned functionality.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Completeness - Displays Programmer Name Program prints out the programmer's name.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Completeness - Gets / Uses User's name Receives input with ReadString. Saves input in a null-terminated BYTE array. Greets user (e.g. "Hello, Username")	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
▶ mpleteness - Displays Introduction Displays program introduction. Program introduction should describe functionality of program.	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Completeness - Prompt for Input Prompts user to enter data, specifying bounds of acceptable inputs.	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts

Criteria	Ratings			Pts
Completeness - Gets data from user Utilizes ReadInt 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. Negative values are rejected.	**3 pts** **Full Marks**	**2 pts** **Partial validation** Validates only one end or neglects to check edge cases.	**0 pts** **No Marks** No validation	3 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
ectness - Number of Terms Correct number of terms are displayed.	**2 pts** **Full Marks**	**1 pts** **Incorrect for small numbers** Correct number of terms for values greater than 3, but fails for one or more of the following values (1, 2, 3)	**0 pts** **No Marks**	2 pts
Correctness - Calculations are Correct	**2 pts** **Full Marks**		**0 pts** **No Marks**	2 pts

Criteria	Ratings		Pts
Correctness - Numbers displayed 4 per line Numbers are displayed 4 per line on sufficiently large console window width. There should be at least 5 spaces between each term. No points granted for implementations that require console window be a specific size to work.	**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 - Solution is non-recursive	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
Upper limit is defined and used as a constant	**1 pts** **Full Marks**	**0 pts** **No Marks**	1 pts
▶ uirements - Well-Modularized Program is divided into logical sections, separated by Section Comment blocks.	**4 pts** **Full Marks**	**0 pts** **No Marks**	4 pts
Requirements - Counted loop implemented with LOOP instruction	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Requirements - Data validation loop is post-test loop	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts

Criteria	Ratings			Pts
Coding Style - 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.	**2 pts** **Full Marks**	**1 pts** **Partial** Some identifiers are named well, with others having no relevance to their functionality.	**0 pts** **No Marks**	2 pts
Coding Style - Readabilty Program uses readable white-space, indentation, and spacing as per the Indentation Style Guide. Logical sections are separated by white space.	**2 pts** **Full Marks**	**1 pts** **Marginally Readable** Program is marginally readable but lacks proper alignment and white space.	**0 pts** **No Marks**	2 pts
(1pt) Extra Credit for columns Fibonacci numbers are displayed in aligned columns.	**0 pts** **Full Marks**		**0 pts** **No Marks**	0 pts
Penalty Remove points here for late assignments. (Enter negative point value, 15% of 'earned' points per day late)	**0 pts** **Full Marks**		**0 pts** **No Marks**	0 pts
				Total Points: 50