Algorithm Design

LAB 09 : NUMERICAL SEQUENCE DESIGN

Due Saturday at 5:00 PM MST

This week, we will create a design for an algorithm to compute a numeric sequence.

Program Description

François is a bit of a rebel. Rather than counting 1, 2, 3, 4, 5 like normal people, he counts 2, 1, 3, 4. "You are crazy" exclaims his teacher. "Your counting scheme makes no sense!" "Not true," patiently explains François. "It makes perfect sense. You count like this:"

$$F_1 = 2$$

$$F_2 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

Thus the first few numbers in François' way of counting are the following:

```
{ 2, 1, 3, 4, 7, 11 }
```

Of course, this can be computed recursively, but that is an $O(2^n)$ algorithm:

```
francois(num)
   IF num == 1
      RETURN 2
   ELSE IF num == 2
      RETURN 1
   ELSE
      RETURN francois(num - 1) + francois(num - 2)
```

Your François algorithm must be much faster to get full credit. Please solve this problem yourself; do not look up a solution on the internet.

Assignment

To submit this assignment, three things are needed: a pseudocode program, the algorithmic efficiency, and a program trace. This will be submitted through I-Learn as a **single-file PDF**.

As with two weeks ago, please use the "Comments..." field to answer the following questions:

- How long did it take for you to complete this assignment?
- What was the hardest part of the assignment?
- Was there anything unclear about the instructions or how you were to complete this lab?

Pseudocode Program

Your program must do the following:

- Prompt the user for a number, representing which François number in the sequence we are to compute.
- Compute the corresponding François value.
- Display the value on the screen.

For this assignment, our job is to create a pseudocode design of the algorithm. You are not to use a recursive algorithm, including any "inspiration" from a similar algorithm in CSE 111. If you are having trouble figuring out how this sequence works, please ask your teacher rather than doing any "research" in the internet.

Algorithmic Efficiency

You are required to compute the algorithmic efficiency of this sorting algorithm. Both name the efficiency (such as O(log *n*)) and give a rationale as to why it is what you say it is. The best solutions would have a line-by-line similar to the examples in the textbook.

Program Trace

Please also create a program trace of your algorithm. Your program trace is to include a single test case: the seventh François number.

Assessment

Your grade for this activity will be according to the following rubric:

	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%
Efficiency 20%	It is unambiguous that the correct algorithmic efficiency was determined	Efficiency determination and rationale are correct	Insufficient rationale or incorrect efficiency	There exists an informal discussion of the algorithmic efficiency	Algorithmic efficiency was not computed for this problem
Trace 30%	The trace is correct	The trace is correct except for one or two minor errors	One major error occurred or the trace is not detailed enough to be helpful	An attempt was made to trace the output, but it did not follow any of the guidelines	No program trace exists
Design Quality 40%	The most elegant and correct solution is found	The design completely covers the problem definition	One aspect of the problem definition is missing or one aspect of the design will not work as expected	Elements of the solution are present	Pseudocode is missing or the provided solution does not resemble the problem definition or a solution was copied from the internet
Professionalism 10%	Professional, beautiful, elegant, single-spaced, using a fixed-width font	Everything is clear and legible, pseudocode used correctly	Misspelling, smudge, incorrect pseudocode, or examples of unprofessionalism	At least one aspect of the design is too messy to read or is not pseudocode	Difficult or impossible to read