Assg 05: Binomial Coefficient.

CSci 515 Spring 2015

2015-02-11

Dates:

Due: Tuesday February 24, by Midnight

Objectives

- More practice writing C functions
- Learn about implementing algorithms using iteration and recursion.
- Practice writing a recursive function.
- Become more familiar with implementing mathematical series on a computer.

Description

In this assignment we will write a recursive function to calculate what is known as the binomial coefficient. The binomial coefficient is a very useful quantity, it allows us to count the number of ways of selecting i items out of a set of n elements. For example, if we have 3 items `A, B, C`, there are 3 ways to choose 1 element from the items: choose A, or choose B or choose C. There are also 3 ways to choose 2 elements from the items: AB, AC, BC. There is only 1 way to choose 3 elements form a set of 3 items: ABC. When we choose 2 elements from a set of 3 items, we normally speak of this as 3 select 2, and mathematically we write this as a binomial coefficient

$$\binom{3}{2}$$

Perform the following tasks:

1. Write a function called `nthFibonacciIterative()`. This version of the function takes a single integer as input, and returns an integer as its result, as already mentioned. You should implement this function using a loop (when needed) to calculate the n^{th} Fibonacci number.

2. Write a second function that will also calculate the n^{th} term in the Fibonacci sequence. But your second function will do this using a recursive implementation. Your recursive implementation should be called `nthFibonacciRecursive()`. You should not use any loops in this version of the function, you will calculate the n^{th} Fibonacci number by recursively calling the `nthFibonacciRecursive()` function itself in the appropriate manner in order to calculate the correct term of the sequence.

3. In your `main()` function, prompt the user to enter a value for n. You should then call each of your two different implementations, and display the result of their calculation of the n^{th} term of the Fibonacci sequence, which if you implement the two different versions correctly, should always give the same result. Be careful of using large values of n, as the recursive implementation especially may take quite a while to calculate in cases of large n.

Your program output should look something close to the following when I run your program. I have run the program multiple times, so that you can also see some examples of correct output for some larger values of n:

```
$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 0

0 term of the Fibonacci series, using iterative method: 0
0 term of the Fibonacci series, using recursive method: 0

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 1

1 term of the Fibonacci series, using iterative method: 1
1 term of the Fibonacci series, using recursive method: 1

$ assg05
```

```
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 8

8 term of the Fibonacci series, using iterative method: 21
8 term of the Fibonacci series, using recursive method: 21

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 10

10 term of the Fibonacci series, using iterative method: 55
10 term of the Fibonacci series, using recursive method: 55

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 35

35 term of the Fibonacci series, using iterative method: 9227465
35 term of the Fibonacci series, using recursive method: 9227465
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions (`main`, `nthFibonacciIterative`, `nthFibonacciRecursive`) have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Assignment Submission

An eCollege dropbox has been created for this assignment. You should upload your version of the assignment to the eCollege dropbox named `Assg 05 Fibonacci Sequence` created for this submission. Work submitted by the due date will be considered for evaluation.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 25+ pts. Your program must have the 2 required named functions, that accept the required input parameters and return the required values (if any).

3. 25+ pts. Your iterative implementation must use loops/iteration to implement its calculation. The function must of course correctly compute the n^{th} term of the series.

4. 40+ pts. Your recursive implementation must perform its calculation using recursion. You must have the correct base cases defined. Your function must of course correctly compute the n^{th} term of the series. trials, and count up the successful trials from all of the trials performed, and return the correct probability ratio. Your ratio must be correct.

5. 10+ pts. You must prompt the user for n in main, and correctly display the returned results form your function as shown.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the assignment to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits.

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening { and closing } for control statement blocks on their own line, indented correctly for the level of the control statement block.

Failure to conform to any of these formatting and programming practice guidelines for this assignment will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.