**Recursion Lab**

**CSC/EGR 1054**

**Spring 2021**

**100 points**

**(5 points) Step 1:** Write a main program that calls the following method with a number of the user’s choice, and prints the result (main and this method should be in same file).

public static int mystery (int num)  
{  
  if (num<=2)  
    return 1;  
  else  
    return mystery(num-1)+mystery(num-2);  
}

**(5 points)** Try the program entering numbers 5 through 7 and make sure the results agree with your trace from class.

|  |  |
| --- | --- |
| num | result |
| 5 | 5 |
| 6 | 8 |
| 7 | 13 |

**(10 points)** Add a method to your program that calculates the same thing iteratively. Have your main call both methods with the following numbers: 7, 14, 35, 40, 42, 46. **Make sure to call the iterative version first** and watch for the results. Enter them in the table below:

|  |  |
| --- | --- |
| num | result |
| 7 | 13 |
| 14 | 337 |
| 35 | 9227465 |
| 40 | 102334155 |
| 42 | 267914296 |
| 46 | 1836311903 |

**(10 points)** What do you observe about the time it takes to do the calculations iteratively compared to recursively?

Iteration is almost instantaneous when it comes to computing 46. But the recursion function takes much longer in time than iteration does.

**(30 points)** What is your explanation?  You should gain some insight from the tree you drew in class. It is more than just the time it takes to create the stack of methods!

By computing the numbers iteratively, it uses a for loop searching through an array to move from the beginning to the end. Iteration will loop through the array until the condition is fulfilled. In this case, we asked the iterative method to take in the array and loop through it. If it is equal to 0 or 1 set the current array index to 1. Then if it is neither 0 or 1, take the index, subtract 1 and subtract 2 and add those two numbers together. Then return the final number based off the initial numbers. So in this case, looping iteratively is faster because it is only doing one pass through which speeds up the process.

Computing by recursion is a slower process because of the functions that are used. In the recursion method, we are asking it to take in a number, if the number is less or equal to 2, return 1. If it is greater than 2, subtract 1 and add it to subtract 2 and return. But in this case of recursion, it is calling the method when the function is applied. So it is looping through over and over, sometimes hundreds of times just to fulfill the initial condition. This takes place all the way through the numbers downward until the number equals 2 which causes this method of computing numbers to happen much slower, but in some cases recursion is much more efficient than iteration.

**(40 points)** Create a recursive method to count the number of 0’s in an array.