C Sharp Programming Exercise 02
Recursive Methods

C# Step by Step

This activity consists of four programming exercises. The following exercises are open book and open note. You are free to use any written documentation you wish. However, these are individual exercises, and you cannot consult with each other in writing your programs.

This programming exercise has four parts consisting of four requirements. The grade for each requirement is indicated, for a maximum of 100 points. At a minimum, your program must compile successfully and run.

A *recursive* function is a function that calls itself. Recursive activities are extremely common in every day life. When you mow your lawn, you cut one mower-width and see if you are finished. If so, you stop and drink a beer. If not, you cut one mower-width and see if you are finished, again and again. When you shop for groceries, you place an item in your cart and mark it off the list. If it's the last item, you check out. If not, you place an item in your cart and mark it off the list, again and again. Recursive functions work exactly the same way. You call the function, perform one task, and check to see if you are done. If so, you stop. If not, you call the function again. I have completed the first part of this exercise below, and you can see how simple this is.

Sum of numbers: beginning code Create a console application that will accept ten numbers between 0 and 100, and report their sum. The program below illustrates one way in which this can be done. Create this program and make sure you understand how it works.

```
using System;

namespace sum2ten
{
    class Program
    {
        static void Main(string[] args)
        {
            int start = 0;
            int end = 10;
            int sum = 0;
            sum = get\_sum(start, end, sum);
            Console.WriteLine(\$"The sum is {sum}");
        }

        private static void get\_sum(int start, int end, int sum)
        {
            Console.WriteLine(\$"get\_sum({start}, {end}, {sum})");
            start = start + 1;
            sum = sum + start;
            if (start < end)
                return(get\_sum(start, end, sum));
            else
                return(sum);
        }
    }
}
```

Assign letter grades: setup Create a method that will accept a numeric (integer, float, or double) argument and return a letter grade: 90 or better is A, 80 or better is B, 70 or better is C, 60 or better is

D, and anything less than 60 is F. You will use this method to assign letter grades in the next parts of this exercise.

Average ten scores: 70 points Create a console application that will accept ten test scores between 0 and 100, average them, and report the numerical grade. For example, a teacher will input ten test scores and compute the average numerical grade. Assign a letter grade to the student.

Average a specific number of scores: 80 points Create a console application that will accept an arbitrary number test scores (as specified by the user) between 0 and 100, average them, and report a numerical grade for the average. For example, a teacher will input the total number of tests, then input the specified number of test scores and compute the average numerical grade. Assign a letter grade to the student.

Average a non-specific number of scores: 90 points Create a console application that will accept a number test scores (as calculated by the number of scores actually entered) between 0 and 100, average them, and report a letter grade for the average based on the usual scale. For example, a teacher will input any number test scores, and compute the average numerical grade and the letter grade.This part required you to program a *stop value*. You can choose any kind of stop value you want, typically stop values consist of “quit,” “exit,” Ctl-C, or Escape. I chose a negative one (-1) as a stop value, assuming that no student would ever score a negative grade on a test. Assign a letter grade to the student.

Calculate ϕ: 100 points The Fibonacci series consists of the integers where each subsequent integer is the sum of the two preceding integers, starting with (1, 1). For example, here are the first nine Fibonacci numbers: (1, 1, 2, 3, 5, 8, 13, 21, 34). In the Fibonacci series, the last number divided by the next-to-last number is the value of phi (ϕ) that you calculated in your first test. The longer the Fibonacci series, the closer you get to the true value of ϕ. For this series, $\frac{32}{21}$ is 1.619.

Your assignment is to calculate the first 40 Fibonacci numbers *using recursion*, that is, using only the if/else construct. You may NOT use iteration (while, for, or do loops). When you reach the 40th Fibonacci number, do the calculation and print out the value of ϕ..

Here is the expected output. I used the `long` data type to get 90. All you have to get is 40.

```
PART 4
1. 1 + 1 = 2
2. 1 + 2 = 3
3. 2 + 3 = 5
...
87. 679891637638612258 + 1100087778366101931 = 1779979416004714189
88. 1100087778366101931 + 1779979416004714189 = 2880067194370816120
89. 1779979416004714189 + 2880067194370816120 = 4660046610375530309
90. phi is 1.61803398874989
Press any key to continue . . .
```