C# Programming Homework 04

Chapter 04, C# Step by Step

Readings

Read chapter 4 in the *C# Step by Step* book.

Discussion Questions

Answer the discussion questions in writing for chapter 4.

1. What are all possible values of *Boolean expression*? True/False
2. List the equality operators. List the relational operators. List the logical operators. How are they the same? How are they different? NOT, ==, !=; Logical: <, <=, >,>=
3. What is the general concept of *short circuiting*? This question has a short and simple answer and you do not need to have a detailed response. If the beginning of an 'and' statement is false, you don't check the others. If the 'or' is true you don't need to check the rest of the statement. && (and), || (or)
4. What are the difference in how *short circuiting* works for `&&` and `||`? both are true, or at least one is true
5. Look at the list of operators. What operator has the highest precedence? () Which has the lowest? "=" (the assignment operator)
6. In an *if* or *else* construction using multiple lines of code, what effect does the use of curly braces have? hey make the code in the braces part of the if/else statement. Otherwise, only the first line would be recognized as part of the 'if' statement.
7. In a *switch* statement, what happens if you omit *break*? The code will 'fall-through' and execute the rest of the program.
8. The four keywords in a switch statement are *switch*, *case*, *break*, and *default*. Explain what each keyword does. BREAK breaks the flow of the code & returns control to outside code. DEFAULT is code the switch executes.; SWITCH is the keyword for initiating a 'switch' statement.; CASE is a label to compare the expression.
9. Look at the source listing below. It contains two methods: `recurr1()` and `recurr2()`. There is a significant difference between the two methods. What is it? Recurr2 is 'tail recurssion'. The braces create a "stack" memory location for the operation of the method. The stack unwinds when the condition returns true.
10. (Not in book) What is a *recursive* method? Using a language you know (such as English), write a recursive method that adds up the integers in a list of integers. The input to the method is a list of integers and the output is a scalar value representing a sum.
11. (Not in book) Read a short summary of *De Morgan's* laws.
    (a) Explain how this statement, "It's not snowing or raining," is the same as this statement, "It's not snowing and it's not raining."
    (b) Explain how this statement, "I'm not running and walking," is the same as this statement, "I'm not running or I'm not walking."

```
using System;

namespace Recur_ch03_text
{
    class Program
    {
        static void Main(string[] args)
        {
            int initial = 5;
            Console.WriteLine($"Calling␣recurr1({initial})");
            int recur = recurr1(initial);
            Console.WriteLine($"recurr1␣is␣{recur}");
            Console.WriteLine($"Calling␣recurr2({initial})");
            recur = recurr2(1, initial);
            Console.WriteLine($"recurr2␣is␣{recur}");
        }

        private static int recurr1(int initial)
        {
            if (initial <= 1)
                return 1;
            else
                return initial * recurr1(--initial);
        }

        private static int recurr2(int product, int initial)
        {
            if (initial <= 1)
                return product;
            else
                return recurr2(product * initial, --initial);
        }
    }
}
```