

General Instructions:

You are to utilize the template, source code files provided on the Canvas website for this course. You must complete all the required sections in the comment header blocks in the template files.

You will be creating a C++ program that demonstrates the implementation of two recursive methods, based on the specification described in the text, Chapter 8, *In Practice*, Problem 8.19 on page 316. The purpose of each recursive method is to return the number of “1”s in the value supplied to each of the required recursive methods.

A Client program is supplied with this assignment for you to use to test your solution. An additional file containing the sample run of that Client program with the instructor’s solution to the assignment is also supplied. **Specific Instructions:**

1. Recursive Method #1

- This version of the method will process a *const string* object that contains just “1” an “0” characters. The method will return the number of “1” characters in the *string* object.
- This version will require a “Client” interface that will, 1) check all characters in the *const string* object passed to it by reference to make sure they are **only** “1”s or “0”s ,and 2) invoke the recursive method to include an *index* value to the “string” object for analysis.
- Since individual characters of the string object need to be evaluated, a Client interface method must be used to, 1) validate the string object, see above, and 2) invoke the recursive method to determine the number of 1s, starting at the first character.

2. Recursive Method #2

- This version of the method will receive a *const unsigned int* value, initialized as *hex* value. For example:

```
const unsigned int VALUE = 0x5ac934; // 0101 1010 1100 1001 0011  
0100
```

 Note: The equivalent, decimal value: 5949748
- The User is allowed to enter values, as *hex* values and the number of “1”s are displayed.

3. Bonus (5 points)

- For Method #2, implement it given the *hint* described in the text.
- **Source Code file, BinaryOnes.cpp**

Sample Run:

Lab #4: Recursion

This program demonstrates two versions of counting the number binary "1"s in, 1) a "string" of "1" and "0" characters, and 2) "unsigned int" values expressed as "hexadecimal" values, using recursive methods. For version 2), a fixed value is submitted for analysis and the User is allowed to enter multiple "hex" values for analysis.

Example #1:

Given:

```
int nمبرOnes;
const string BINARY_STRING_VALUE("100110100111010101101");
nمبرOnes = binaryOnes(BINARY_STRING_VALUE);
cout << nمبرOnes;
```

Result:

Number of "1"s: 12

Example #2:

```
Given: const unsigned HEX_VALUE = 0x5ac934; // 0101 1010 1100
1001 0011
0100
```

```
nمبرOnes = binaryOnes(HEX_VALUE);
cout << nمبرOnes;
```

Result:

Number of "1"s: 11

Example #3:

Enter a value as a hexadecimal value (0xDDDD): 0x1234

Given:

```
"value: 0x1234: 0001 0010 0011 0100
```

```
nمبرOnes = binaryOnes(UsersValue);
cout << nمبرOnes;
```

Result:

Number of "1"s: 5

Evaluate another value(y/n)?: