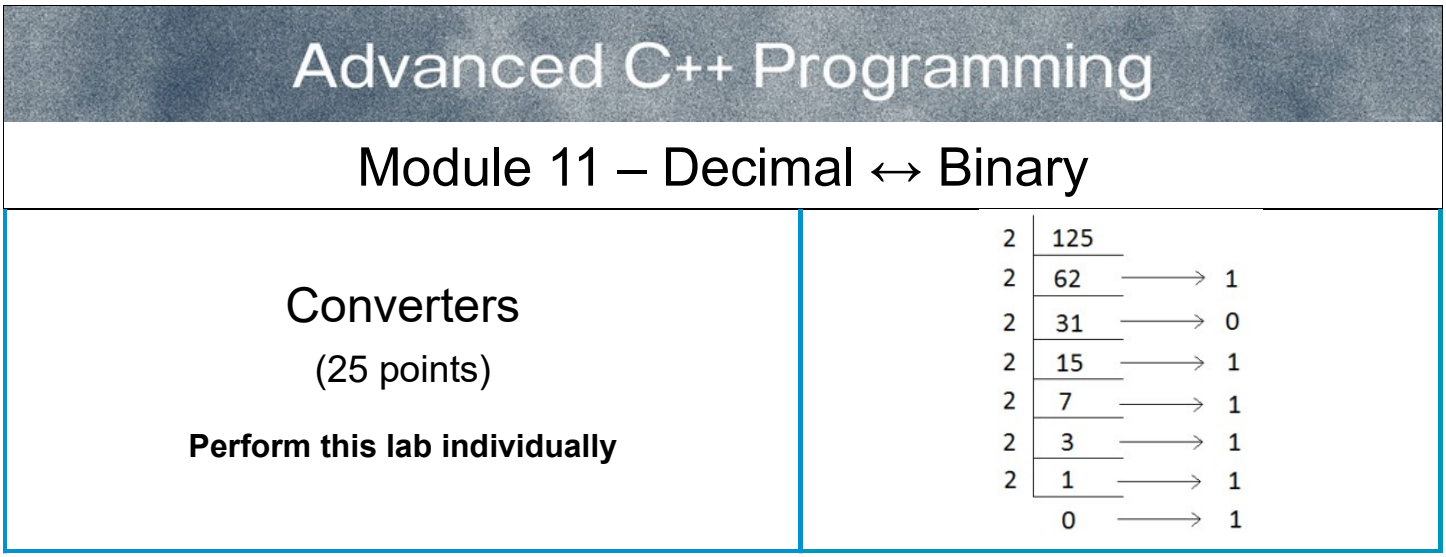
Lab 11 - Recursion

Start Assignment

Due Monday by 11:59pm **Points** 20 **Submitting** a file upload



Summary

In this lab, you will create several functions to convert between base 2 and base 10 numbers

- Part 1: Convert several decimal numbers to their binary (string) equivalents.
- Part 2: Do the reverse of Part 1.

In each part, please create an iterative version and a recursive version.

As always, please place all functions after the **main** function in your program file.

Project 1 – Decimal-to-binary converter

Create two functions (one recursive, one iterative) which take a non-negative whole number (in decimal as specified in your C++ source code) and convert it to a **string** of 0's and 1's representing the same number in binary (base 2). *Complete these functions without passing any additional parameter*s.

For example, define an **int** variable and initialize it to (decimal) **100.** Your recursive and iterative functions, when passed this variable, should both return the **string "1100100"**.

Please use only arithmetic operators such as multiplication, integer divide */*, and the modulo **%** operators in this lab. *Do NOT use the* ***pow*** *function or a* ***stringstream*** *object to perform*

conversions or raise numbers to a power, and avoid using the **<<** *and* **>>** *operators.*

An example: To convert the decimal number 100 to a string of 0's and 1's, continuously take the modulo (remainder) of the number when divided by 2, and put those 0's and 1's together to form a **string**:

100 / 2 is **50**, and 100 % 2 is **0**
50 / 2 is **25** and 50 % 2 is **0**
25 / 2 is **12** and 25 % 2 is **1**
12 / 2 is **6** and 12 % 2 is **0**
6 / 2 is **3** and 6 % 2 is **0**
3 / 2 is **1** and 3 % 2 is **1**
0 and 1 % 2 is **1**

The base case is reached when the division by 2 results in a 0

The result is the modulos put together to form a **string** (reading up from the bottom) **1100100**.

To test your program(s), print the binary string equivalents of 0's and 1's for the following decimal numbers: 0, 5, 32, 240, and 682. Both functions (iterative and recursive versions) should take a single **int** as its lone parameter, and return a string. Do not use global variables in your solution.

Project 2 – Binary-to-decimal converter

Perform the reverse of project 1, that is, create two functions (one recursive, the other iterative) which take a **string** of 0's and 1's, and return an **int** with its decimal equivalent. *Again, complete these functions without taking any additional parameters*.

For example, your program should take the **string "1100100"** as input and return an **int** with a decimal value of **100**.

Please use only arithmetic operators such as addition, multiplication, integer divide */*, and the modulo **%** operator in this lab. *Do NOT use the* ***pow*** *function or a* ***stringstream*** *object to perform conversions or to raise numbers to a power, nor use the* **<<** *or* **>>** *operators.*

An example: To convert the **string "1100100"** to a decimal number, look at the following sequence:

1. Set the variable **sum** to 0.
2. Convert the first character in the **string** "1" to a number (i.e., 1) and add to **sum**
3. Multiply **sum** by 2 and add the second character converted a number (2 * 1 + 1 = 3)
4. Multiply **sum** by 2 and add the third character converted to a number (2 * 3 + 0 = 6)
5. Multiply **sum** by 2 and add the fourth character (2 * 6 + 0 = 12)
6. Multiply **sum** by 2 and add the fifth character (2 * 12 + 1 = 25)

7. Multiply **sum** by 2 and add the sixth character (2 * 25 + 0 = 50)
8. Multiply **sum** by 2 and add the seventh character (2 * 50 + 0 = 100)

Voila! The number in **sum** (100) is **"1100100"** converted to a decimal number. Notice that this is the reverse of the conversion example used in problem 1 above.

Do you see a pattern? Can you create an iterative function which converts this pattern into a loop? That will be your *iterative* solution.

For your *recursive* solution, use the same questions you answered above:

- what is the stopping (or "base") case?
- what is the recursive case?
- what will your function take as its parameter(s)
- what will your recursive function return?

Each function (recursive and iterative) should take a **string** as its lone parameter and return a number (an **int**). Do not use global variables in your solutions.

Use the **string**s which you created in Project 1 as test data sets. The results should be the original numbers used as test data in Project 1.

Links

Additional Files and Programs	Next Lab
none	Lab 12 - Polymorphism, Virtual Functions
Homework Assignment	**Prior Lab**
Homework 11	Lab 9 - Advanced I/O