# CSC236 L5: Binary Addition

## This is a lab designed for individual or pair programming.

---

Note that this lab was created by Drs Mario Nakazawa and Jan Pearce of Berea College

---

- **This lab may be done individually or in a team of 2 students (i.e with no more than one partner)** and you may discuss the work with the professor, the TAs, and even your classmates. **However,** making any copies of your work or allowing anyone else to copy your work is and will be treated as **academic dishonesty**.
- You will be committing your source code on Github Classroom and submitting this document to Moodle.

| Enter your name(s): |
|---|

---

## Objectives

- Learn more about binary numbers
- Work with the linked list data structure on an interesting problem
- Be able to understand pre-existing program code and enhance it.

---

## Setup

If you are working with a partner, make sure to fill out the following table as to who is doing what.

| **Team Roles** | **Member Name** |
|---|---|
| **First Driver/Second Navigator** | |
| **First Navigator/Second Driver** | |

## Some Background: Number Systems

We are all familiar with the **decimal** number system we use. The word decimal derives from the union of the Indian Sanskrit word *dasha* meaning 'ten' and the English word *numeral* meaning 'symbol representing a number'. Why ten? Well, humans have 10 fingers! Thus we say that our numbering system is base 10.

Analogously, computers use **binary** as their primary number system, where the word *binary* derives from the Latin word *biniarius* meaning 'two together'[1]. This numbering system is thus **base 2**, and it is appropriate for traditional computers because they have components that are either on and off states back when the switches were mechanical. Today high and low voltages are used.[2]

Each whole number has various representations. For example, the number eleven is represented by 11 in decimal and by 1011 in binary.To understand this, we need to first revisit decimal numbers.

In order to better understand the binary number system that computers use as their primary number system, we will first discuss the decimal system that we use daily because both of these number systems are based upon the idea of **place value**.

### The Decimal Number System

The main principle of the decimal system is that the ten numbers 0 through 9 that can be represented by a single symbol (called **digits** in base 10) are simply represented by their own digits. What if we want to go one MORE than 9, the largest single symbol value? The number ten is considered as a new unit because we have the symbol "1" in the ten's place followed by the smallest single symbol value, "0" in the one's place. By cycling through this process, we can represent numbers 10 through 99 using only two digits. What happens if we want more than 99? The cycle starts again at a larger place value.[3] We can show the growth of the place value visually in this table, where the place values increase we move to the left of the decimal point and vice versa to the right. Note the exponent at each place.

| Place Value | ... | $10^4$ | $10^3$ | $10^2$ | $10^1$ | $10^0$ | . | $10^{-1}$ | $10^{-2}$ | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| Decimal Value | | 10000 | 1000 | 100 | 10 | 1 | | 0.1 | 0.01 | |

We can compute the decimal integer 10,574 as the sum of each digit multiplied by its place value:

[1] https://www.etymonline.com/word/binary

[2] The advent of quantum computers has complicated things by introducing the idea of **qubit**, or *quantum bits*. You can look further into this fascinating field (https://www.epiqc.cs.uchicago.edu/qc-introduction/). *Focus on this lab, though!*

[3] This way of counting is very old. Indo-European languages, which are spoken from India to Europe, all have the same counting system as well as very similar words for numbers. We can conclude that the basic Indo-European mother language had a similar method of counting - called decimal counting as early as 3000 BC. The origins of the decimal counting system are somewhat hidden, but we can imagine the spread of this system through trade routes.

$$10{,}574 = (1 \times 10^4) + (0 \times 10^3) + (5 \times 10^2) + (7 \times 10^1) + (4 \times 10^0)$$

### What about the Binary Number System?

A computer system can be modeled such that a computer circuit contains components where '0' represents a low voltage (sometimes called "off") and a '1' as a high voltage (sometimes called "on".) By manipulating the circuits of the computer system to behave like the binary number system, the computer is able to perform calculations, which is the basis of all its operations. Each 0 or 1 value is stored in a BInary digiT, often referred to as a **bit**.

A binary numbering system also uses place values except that each place represents a power of 2, just as the place represents a power of 10 in the decimal system. If we modify the table above and replace all 10's with 2's, we get something that look familiar:

| Place Value | ... | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | . | $2^{-1}$ | $2^{-2}$ | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Decimal Value | | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | | 0.5 | 0.25 | |

An 8-bit number is called a **byte** inside a computer. A byte such as 11010010 can be rewritten in decimal by using the place value as shown in the table above:

$$11010010 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

A 4-bit number (called a nibble[4]) can hold 16 values because its largest place value is $2^4 = 16$.

| nibble value | decimal value | nibble value | decimal value |
|---|---|---|---|
| 0000 | 0 | 1000 | 8 |
| 0001 | 1 | 1001 | 9 |
| 0010 | 2 | 1010 | 10 |
| 0011 | 3 | 1011 | 11 |
| 0100 | 4 | 1100 | 12 |
| 0101 | 5 | 1101 | 13 |
| 0110 | 6 | 1110 | 14 |
| 0111 | 7 | 1111 | 15 |

**If you want some more practice, try this game: Binary Numbers Game**

[4] Computer scientists can be so funny! https://en.wikipedia.org/wiki/Nibble

## The Single Bit Adder (A+B)

In computer science, the basic adding component is a **single-bit adder,** which operates as:

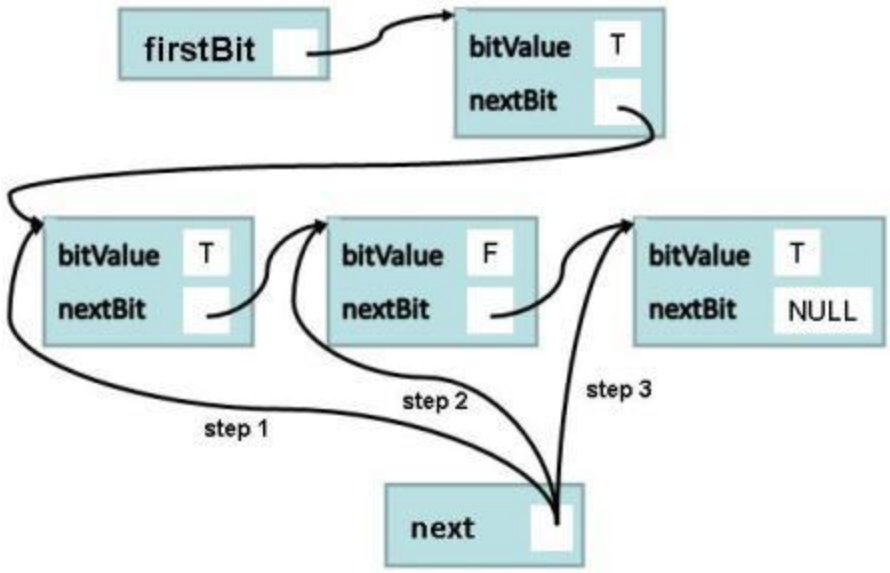
0+0=0
1+0=1
0+1=1
1+1=10 (two in binary is a 1 in the 2's place and a 0 in the 1's place.)

If we consider the inputs labeled as A and B, this adder generates a single-bit output for the sum except for the case where the inputs are BOTH 1. The sum bit, which is the rightmost bit, is a zero because 10 in binary is 2 in decimal. The "1" bit to the left is called the carry and is then added to the sum in the next place.

## Representing Longer Binary Numbers

Representing these binary numbers in the computer can be done using linked lists, where each link holds a binary value (0 or 1) and a link to the next bit in the number. In the code that is provided for you, the bit value is stored as a Boolean value, where the zero corresponds to **false** and a one value is **true**. This choice is more efficient than storing the one of zero as an integer.

Starting with the least significant bit, the number 1011 (or T F T T in Boolean) can be created to produce a pictorial representation below.




Notice that the linked list is "backwards", where the more significant bits are further down the list, so 1011 is stored backwards as → T → T → F → T → NULL.

## Milestone 1 - Understanding the Codebase

**A large part of the implementation has been completed for you. You must not change the parts of the classes that are already completed and tested! You are ONLY to ADD to the codebase, you must not change the existing codebase.**

Clone the repository **L5: Binary Addition** for the starter files. Remember that you cannot reuse team names, so it is recommended that you include “L5” in your team name.

1. **CODE REPOSITORY**: Paste a link to your GitHub Repository here. Note that you MUST begin with and correctly use the starter code.

The starter code contains the following in C++:

- a **`Bit`** class (This class may only be changed by adding a single method.)
- a **`BinaryNumber`** class that uses a linked list to represent a binary number (This class may not be changed at all.)
- a **`BinaryNumber`** driver main program (which will need changing)

These files already do the following correctly:

1. Provides an instance variable for the first link in the binary linked list (implemented as a **`Bit`** object)
2. Takes a decimal number and converts it to binary, storing the binary number in a BinaryNumber object which is a linked list.
3. Takes a BinaryNumber object and converts it to decimal.
4. Outputs the binary number
5. Provides the accessor functions for the first node of a linked list

In addition, the program has a `main()` driver function that tests the **`BinaryNumber`** class.

### Your task

Your task is to write an increment method as part of the BinaryNumber class that correctly increments (adds 1 to) the linked list version of the BinaryNumber. Note that you must do this with the provided linked list. No credit will be given for other methods for incrementing because this lab is designed to help you to understand linked lists.

---

### Reflection Prompts to Be Submitted with First Milestone:

Included with the assignment are implementations of the BinaryNumber class in C++ in which binary numbers are represented using linked lists bit nodes for each bit of the number. Each node in that list

is essentially a bit of that number, and adding one to that number can result in a "ripple effect" of carries, which can increase the length of the list.

1. **Bit Class:** Describe in your own words what the purpose of this class is.

2. **Binary Number Class**: Describe in your own words what the purpose of this class is.

3. **Main Program:** Describe in your own words what the primary purpose of this program is.

4. **Creating a new node:** Identify all lines of existing code that create a new node in the BinaryNumber linked list.

5. **Iterating through BinaryNumber:** Identify **all** lines of code that advance a pointer from one node to the next node. Be sure that you identify all of these.

6. **Find and Explain Iteration in the Provided Codebase**: One or more of the methods in the provided codebase iterate through the linked list. Identify all of them and explain.

   a. how they move from node to node, and
   b. how they decide when to stop iterating.

7. **Arrow Notation**: Explain the significance of the arrow (`->`) notation for the nodes of the C++ linked list.

8. **Linked List Order**: The binary representation has the head of the linked list as the least significant bit (i.e. the number 13 in base 10 is represented as `->1->0->1->1->NULL`.) Identify one significant advantage this representation has with regards to how this number can be incremented.

9. **High-level Whiteboard Design Plan**: Note that you MAY NOT directly use **ANYTHING** other than ideas from the `BinaryNumber` class because the entire purpose of this assignment if for you to learn to use the a given linked list data structure, so you MUST write a method in the that iterates through the given linked list using self-contained code. Do not use ideas from the internet.

   Write a high-level whiteboard design for the increment method that you will write. Note that the details are not necessary here, just the big ideas of the algorithm.

10. **Other Representations**: There are typically many design choices that could be made in terms of data structures that can be used for a given task. Identify two other possible data structures that one might alternatively use to represent a binary number. For each, determine whether you think that adding would be easier to write and/or more efficient in terms of Big O and why.

11. **Least Significant Bit**: Notice the `convert_decimal_to_binary()` method refers to the `leastSignificantBit` instance variable only in the first half of the method, and the rest referred to `bitRef`. Explain why the method still works even though it does not refer to `leastSignificantBit` all the time.

12. **Assigned Tasks (Only optional if working alone)**:
    List who will implement each of these (components of the main and each class method.) Be careful to keep this equitable. It is NOT okay to have one person do the majority of the work because they also then do the majority of the learning.

## Milestone 2 - Implementation

Your job is to design and implement a single function called `increment()` that increases the number represented by a **`BinaryNumber`** object by one. We cannot emphasize this fact enough: You must do this in the provided class by using the linked-list using the provided binary representation correctly. **No credit will be given for work that does otherwise. In other words, you will not get credit for creating your own version.**

**IMPLEMENTATION REQUIREMENTS**:

Your program must have good structure and style and the following:

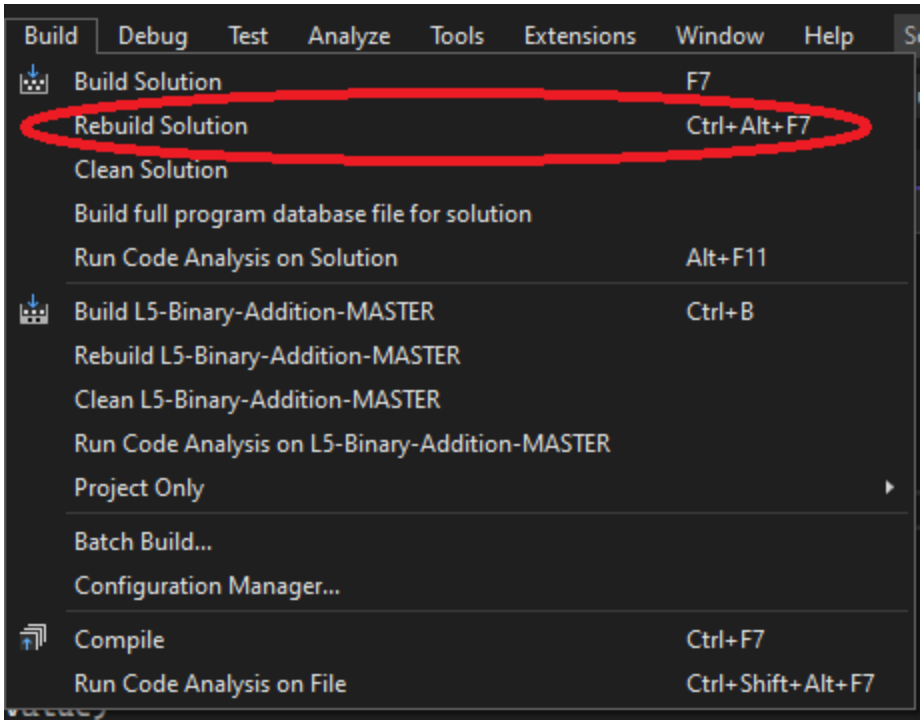
- Add “L5”, a description of this lab, and your name(s) as authors at the top of the file with the `main()` function.
- Include a descriptive header as a comment at the top of your source code which include the names of all authors, the lab name and purpose, and a statement that this is the original work of the author except where specifically documented.
- Design in a modular fashion, correctly using functions for each task with correct parameter passing and appropriate use of returns.
- Use only meaningful variable and function names.
- Implement the `increment` method using the given linked list in the **`BinaryNumber`** class. Do NOT change the classes except as indicated.
- Important Note: You MUST work with the binary as represented by the `BinaryNumber` class. You may NOT just work with decimal and use the code we have given you to convert. The primary purpose of this lab is for you to learn to work with the linked list data structure.
- Include descriptive docstrings for each function, class, and method that you write. Even better, each function and method should have appropriate pre- and post-conditions.

**You must make sure to use the linked list data structure (change the bitValue values and traverse the linked list on the ripple carry) for this lab.**

### Some complications:

- You must handle "overflow". For example, suppose that the number in the object is "111", which is 7 in decimal. Thus incrementing it would not only cause the carry to ripple 3 times, it would necessitate expanding the linked list to store the bit that got added to the end ("111" + "1" = "1000"). In other words, there were 3 nodes before the number was incremented, but it has 4 nodes afterwards. In terms of the linked list implementation, → T → T → T → NULL becomes → F → F → F → T → NULL.
- Be sure to consider and handle what incrementing an empty binary number linked list should do.
- **Important note:** It is recommended that you **Rebuild your Solution** if you get weird errors at runtime or if you don’t seem to see your changes. This is because when working with pointers,

you can mess up things in memory. Rebuilding is very easy and fast to do in Visual Studio Community, so if your logic seems correct but the output makes no sense to you, **Rebuild your Solution!**




## To Submit

- **Milestone 1**: Download this document, complete the "Milestone 1" section, and upload it to Moodle by the Milestone 1 due date and time. This section consists of the first six prompts. Pull your repo, make at least one commit.
- **Milestone 2**: Make a sincere attempt to complete your program so that you can ask questions. Ensure you sync your repo to GitHub. Some parts may not be working. **Hint**: Don't forget the important note about rebuilding your solution during debugging! (See above!)
- **Total Completion**: Complete the implementation and the `README.md` file by replacing all the **`FIXMEs`** with your responses, commit and push your repository to the **main branch** in your Github repository before this project is due, and submit the link to your Github repository in the Moodle textbox. Remember that there are more than 1000 repositories in the csc236 Github organization, so do not expect the TAs to be able to find your repository if you fail to submit the link. Also, do not commit or push anything after the deadline or it will be counted as late.