# CS 570: Homework Assignment 1
# Due: February 15th, 11:59pm

## 1 Assignment Policies

**Collaboration Policy.** Homework will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

**Under absolutely no circumstances code can be exchanged between students.** Excerpts of code presented in class can be used.

**Your code must include a comment with your name** .

## 2 Assignment

Define a class `BinaryNumber` that represents binary numbers and a few simple operations on them, as indicated below. An example of a binary number is

1011

Its *length* is 4. Note that its leftmost digit is the least significant one: it represents the decimal number $1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 = 13$. This is called *little-endian* format. You may use *big-endian* if you prefer; in that case you must state so at the beginning of your code as part of the comment.

This assignment requests that a number of operations be supported. They are divided into two groups. The first is a set of basic operations, the second is slightly more challenging and addresses addition of binary numbers.

### 2.1 Basic operations

The following operations should be supported:

- A constructor `BinaryNumber(int length)` for creating a binary number of length `length` and consisting only of zeros.

- A constructor `BinaryNumber(String str)` for creating a binary number given a string. For example, given the string `"1011"`, the corresponding binary number should be created. For this exercise you will have to use some standard String operations. These are listed in the "Hints" section below.
- An operation `int getLength()` for determining the length of a binary number.
- An operation `int getDigit(int index)` for obtaining a digit of a binary number given an index. The starting index is 0. If the index is out of bounds, then a message should be printed on the screen indicating this fact.
- An operation `int toDecimal()` for transforming a binary number to its decimal notation (cf. the example given above).
- An operation `void shiftR(int amount)` for shifting all digits in a binary number any number of places to the right, as indicated by a parameter `amountToShift`. The new digit should be 0. For example, the result of shifting "1011" 3 places to the right should yield "0001011".

## 2.2 Addition of Binary Numbers

Here is an example of how two binary numbers of the same length are added[1].

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 1 | 1 | | (carried digits) |
| | 1 | 0 | 1 | 1 | 0 | | |
| + | 1 | 1 | 1 | 0 | 0 | | |
| = | 0 | 0 | 1 | 0 | 1 | = 20 | |

Note that it is possible for the addition of two numbers to yield a result which has a larger length than the summands. In that case, this should be flagged by an appropriate boolean data field. Here is such an example.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 1 | 1 | 1 | | (carried digits) |
| | 1 | 0 | 1 | 1 | 0 | | | |
| + | 1 | 1 | 1 | 0 | 1 | | | |
| = | 0 | 0 | 1 | 0 | 0 | 1 | = 36 | |

This data field should be added to the data fields of the class `BinaryNumber`.

Implement the following operations:

- `void add(BinaryNumber aBinaryNumber)` for adding two binary numbers, one is the binary number that receives the message and the other is given as a parameter. If the lengths of the binary numbers do not coincide, then a message should be printed on the screen indicating this fact. Otherwise, it modifies the receiving binary number with the result of the addition.
- An operation `clearOverflow()` that clears the overflow flag.
- An operation `String toString()` for transforming a binary number to a String. If the number is the result of an overflow, the string "Overflow" should be returned.

[1] Source: `https://en.wikipedia.org/wiki/Binary_number`

### 2.3 Hints

- For the `BinaryNumber(String str)` constructor, the following operations might come in handy:
  - `char java.lang.String.charAt(int index)`, which returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.
  - `int java.lang.Character.getNumericValue(char ch)`, which returns the int value that the specified Unicode character represents.
- For the `shiftR(int amount)` operation, it might be useful to define an auxiliary private method `reallocate`, that makes room for a new digit. This operation would in turn use `int[] java.util.Arrays.copyOf(int[] original, int newLength)`, which copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.

## 3 Submission instructions

Submit a single file named `BinaryNumber.java` through Canvas. No report is required. Your grade will be determined as follows:

- You will get 0 if your code does not compile.
- The code must implement the following UML diagram precisely.
- We will try to feed erroneous and inconsistent inputs to all methods. All arguments should be checked.
- Partial credit may be given for style, comments and readability.

| **BinaryNumber** |
| --- |
| private int data[]<br>private boolean overflow |
| public BinaryNumber(int length)<br>public BinaryNumber(String str)<br>public int getLength()<br>public int getDigit(int index)<br>public void shiftR(int amount)<br>public void add(BinaryNumber aBinaryNumber)<br>public String toString()<br>public int toDecimal()<br>public void clearOverflow() |