COP 3530 Data Structure and Algorithm Analysis
Homework 1

Feng-Hao Liu

In this assignment, you are going to write a program for the following tasks in **a single cpp** file:

- **Task 1:** write a function that takes input an integer int n, and returns void. The function should print the binary representation of n starting from the most significant bit. For example, if the input is 26, then the function should print 11010 on the screen. Here you do not need to worry about the problem of overflow.
- **Task 2:** write a function that converts a binary number into a decimal number. In particular, the function takes inputs an array int* p and int n, where $p[n-1], p[n-2], \ldots, p[1], p[0]$ represent the binary digits from the most significant bit. Your function returns an integer that is equal to the number p represents. You can assume that $n \leq 31$ so that there is no overflow. (Most significant bit means the left most bit.)
- **Task 3:** In your main function, you should write some test cases to test your Tasks 1 and 2. The format is flexible.

In the class, we have discussed about the algorithms. Here is a quick recap. Given a decimal number n, we can represent n into the following way:

$$n = a_k 2^k + a_{k-1} 2^{k-1} + \cdots + a_1 2 + a_0,$$

where k is roughly $\log_2(n)$ and all a_i's are binary digits, i.e., either 0 or 1. The binary representation of n should be $a_k a_{k-1} \ldots a_0$. If you are given a binary string $a_k a_{k-1} \ldots a_0$ (represented by an array), then you can compute n from the above formula directly.

The slightly more non-trivial part is to to compute the binary digits from n. Here we can first determine the a_0 by using mod 2. (Think about how and why). Then we know:

$$n/2 = a_k 2^{k-1} + a_{k-1} 2^{k-2} + \cdots + a_1.$$

So we can use the same idea to find out a_1. Proceeding in this way, we can find out all the a_i's.

For more information about binary numbers, you can find more details from Wikipedia (`http://en.wikipedia.org/wiki/Binary_number`).