CS 3432 – Computer Organization

Lab 1 Author: Robert Alvarez

This lab will introduce you to the bitwise operators not (~), and (&), or (|), xor (^), left-shift (<<), and right-shift (>>) through different exercises explained below. Pointers and memory addresses would also be introduced/used for problems 6 and 7; only 7 use heap memory.

**Instructions:**

Assignments must be done in pairs and submitted through GitHub Classroom. Use the following link to access the assignment: <https://classroom.github.com/a/L59oEPNz>

Note that only assistance from your teammates, instructor, TA, or IA will be permitted.

**Bitwise Operators and Introduction to Memory Addresses and Structs**

In the preamble of each function definition, there is a set of parentheses containing bitwise operators, which are mandatory for constructing the function. Failure to employ each of these operators at least once within the function will result in the function’s lack of grading. This requirement is implemented to mitigate the potential for plagiarism to the greatest extent possible.

1. [10 points] (^) Develop a function capable of exchanging the values of two short numbers without necessitating using a third variable.
2. [10 points] (~, &) Given a 16-bit unsigned integer, create a function that returns 0 if the variable contains an even number of set bits (1s) and returns 1 if it contains an odd number of set bits. Your solution must have a time complexity of O(# of set bits).
3. [10 points] Given an integer n and a short k with values between 0 and 15, make functions to
   1. (&, ~, <<) turn off,
   2. (&, <<) check and
   3. (^, <<) toggle

the bit at the specific location.

1. [15 points] (<<, |, &, >>) Design a function that determines whether the binary representation of a given number is a palindrome or not.
2. [10 points] Given a 16-bit number, create a function that returns its bits in reverse (&, |, <<). Your solution must have a time complexity of O(# of set bits).
3. [20 points] (<<, &) Given an array of 120 unsigned characters, each consecutive 12 entries would be used to draw a number from 0 to 9. Make a function that receives the 2D array, one number that represents which number to draw, and an enumerate variable of how to draw it. The enumerate variable is declared as follows:

enum reflex\_num{ NO, x\_axis, y\_axis, x\_y\_axis };

* 1. Number is drawn by printing the binary representation of each number per row,

‘ ‘ when 0 and ‘\*’ when 1.

For example, the entry with value 0x7E would generate “ \*\*\*\*\*\* ”.

1. [15 points] (&, <<, |) Create a function that receives a pointer to a sequence of characters, and it returns 1 if all its characters are unique, 0 otherwise. The function must run in O(n) time and O(1) space. [Hint: Use an array with a constant number of fixed-sized integers (sizes must be maintained regardless of the architecture)].
2. (^) Two files, XORDoubleLL.h and XORDoubleLL.c, are given with the declaration of the double head linked list struct and five functions, where only three are implemented. One is to print the linked list data in traversing forward on the list, one is to insert a node at the beginning of the list, and one is to perform the XOR operation between two addresses. Your job is to:
   1. [15 points] Implement insert\_node(node\_t \*, node\_t \*) function such that it inserts the new node so that the node preceding it possesses a lower memory address, while the node following it has a higher memory address.
   2. [15 points] Implement remove\_node(node\_t \*, unsigned long) function, which removes a node (if it exists) based on its address and updates the linked list accordingly as required.

A rudimentary implementation and a brief explanation of a single-field doubly linked list can be found on the [geeksforgeeks](https://www.geeksforgeeks.org/xor-linked-list-a-memory-efficient-doubly-linked-list-set-1/) website.

**Permitted standard libraries:**

* math.h - log2()
* stdio.h - putchar(), malloc(), free()
* stdint.h - uintptr\_t

No additional standard C library, function, or variable is allowed. If there is a desire to use anything else, please get in touch with the AI or the TA for assistance in finding an alternative approach that does not rely on said library, function, or variable. Failure to adhere to this rule will result in a score of 0 points for the compromised function(s).

**Deadline (GitHub Classroom):** September 21st, 2023, by 11:59 pm.

1. Source code (Only .c and .h files)
   1. If another file that was not given is included, the Makefile must also be modified to for a successful compilation and linkage. Failure to do so would result in a 10-point deduction.

**Grading:** (Total of 120 points where 20 points are extra credit)

Each function has the weight grade value of its implementation.