School of Computer Science
Faculty of Science

COMP-2650: Computer Architecture I: Digital Design
Winter 2021

Lab#	Date	Title	Due Date	Grade Release Date
Lab 11	Week 11	**Binary Codes (Gray Code)**	March 23, 2021 Tuesday Midnight AoE Wednesday 7 AM EDT	March 29, 2021

This lab's objectives will be to master the topics in logic circuit design by implementing the algorithms with a programming language, herein, C/C++.

Step 1. Environment Setup

Our programming environment is the same as the first lab (Lab 01). In this lab, we want to implement the Gray code, named after Frank Gray[1]. This code is an ordering of the binary numeral system such that two successive values differ in only one bit. For example, the Gray codes for the decimal numbers 4 and 5 are 0110 and 0111, where the only change is the first bit. To implement Gray code, we have first to convert a given decimal number to a binary number. Then, we follow the below steps:

Moving from the highest significant bit to the lowest significant bit (last bit to the first bit)
1) The last bit of the Gray code is the same as the last bit of the binary number
2) The i-th bit of the Gray code is the XOR of the i-th and (i+1)-th bits of the binary number

For instance, the Gray code for 20 is:

$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code					
$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1				
$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	$1\oplus0=1$			
$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	$0\oplus1=1$		
$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	$1\oplus0=1$	
$(20)_{10}$	Binary Number	1	0	1	0	0
	Gray Code	1	1	1	1	$0\oplus0=0$

[1] https://en.wikipedia.org/wiki/Frank_Gray_(researcher)

Lab Assignment

You should implement the above algorithm output a menu of commands as follows:

```
Enter the encoding command number:
0) Exit
1) Gray code
```

If a user selects (1), the program asks for a decimal number:

```
Enter a decimal number:20
```

When the user enters a decimal number, the program should print out the Gray code for the given decimal number as shown below:

```
Gray code for 20 -> 11110
```

and come back to the main menu. If the user selects (0), the program ends. Please restrict the user to enter inputs within the range [0,2^8-1=255] for the decimal number. For instance, if the user enters -1, 999, print out an error message and come back to ask for correct inputs.

It is required to write a *modular* program. Please put the part of the code that outputs the Gray code in a new function called `to_Gray()` inside the main.c file.

Deliverables

You will prepare and submit the program in one single zip file `Lab11_UWinID.zip` containing the following two items:

1. The entire project folder `Project` including the code file (`main.c` or `main.cpp`) and executable file (`main.exe` in windows or `main` in mac/linux)

2. The result of the commands in the file `Results.pdf`. Simply take screenshots of the results and save (print) them into a single pdf.

2. [Optional and if necessary] A readme document in a txt file `ReadMe.txt`. It explains how to build and run the program as well as any prerequisites that are needed. *Please note that if your program cannot be built and run on our computer systems, you will lose marks.*

In sum, your final `Lab10_UWinID.zip` file for the submission includes 1 folder (entire project folder), 1 image (results snapshot) and 1 txt (report). *Please follow the naming convention as you lose marks otherwise.* Instead of UWinID, use your own UWindsor account name, e.g., mine is hfani@uwindsor.ca, so,

`Lab11_hfani.zip`

- (75%) `Project`
  - [any required library, header or source files]
  - `main.c` or `main.cpp` => Must be compiled and built with no error!
  - `main.exe` or `main`
- (10%) `Results.pdf`
- (Optional) `ReadMe.txt`

(10%) Modular Programming (using separate functions)
(5%) Files Naming and Formats, and Folder Structure