CS-UY 2214 — Recitation 1

Introduction

Complete the following exercises. Put your answers in a plain text file named `recitation1.txt`. Number your solution to each question. When you finish, submit your file on Gradescope. Then, in order to receive credit, you must ask your TA to check your work. Your work should be completed and checked during the recitation session.

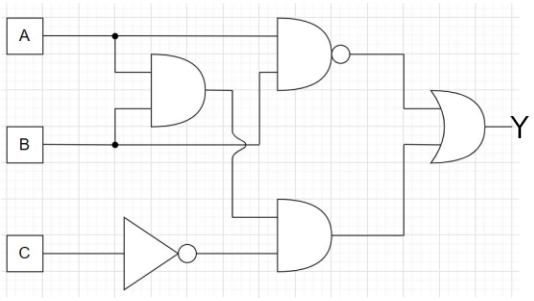
Please note that your solutions must be in a *plain text file*. Other formats, such as PDF, RTF, and Microsoft Word, will not be accepted. Here are some recommended editors that produce plain text files:

- Notepad (comes with Windows)
- TextEdit (comes with Mac OS); note that if you are using TextEdit, you need to select "Make Plain Text" from the Format menu before saving the file
- gedit (available on most Linux distributions)
- nano (available on most Linux distributions)
- Sublime Text
- VSCode
- Atom
- Vim
- Emacs

For questions that require a solution expressed as an image, submit the image as a separate file. The image file should be named `recitation`n`q`m, where n is the recitation number and m is the question number; use an appropriate suffix (either `jpg` or `png`).

Problems

1. Consider the following circuit.



Please note that in the above diagram, an arced line represents a wire crossing another wire without intersecting it. A filled dot on a wire intersects an intersection, i.e. the signal passes to multiple outputs.

Express the circuit as a boolean expression, using only AND, OR, and NOT. Provide a truth table for this circuit.

2. Using only AND, OR, and NOT gates, construct a circuit diagram that will calculate XOR. Your circuit will have two inputs A and B, and one output Y.

   Submit your answer as an image, in accordance with the instructions at the beginning of this document. Your image may be a diagram created in a drawing program, or it may be a photograph of a hand-drawn diagram on paper.

3. Convert the following decimal numbers into binary.

   (a) 35

4. Convert the following binary numbers into decimal.

   (a) 1001

5. Convert the following decimal numbers into hexadecimal.

   (a) 35

   (b) 64

6. Convert the following hexadecimal numbers into binary.

   (a) f00f

   (b) abcd

7. Convert the following decimal numbers into 8-bit binary 2's complement. Write all the digits.

   (a) 34

   (b) -34

8. Convert the following 8-bit binary unsigned numbers into decimal.

   (a) `11010011`

   (b) `00001000`

   (c) `11111111`

9. Convert the following 8-bit binary 2's complement numbers into decimal.

   (a) `11010011`

   (b) `00001000`

   (c) `11111111`

10. Consider the following C++ program. Do not run it.

```
#include <iostream>
using namespace std;
int main() {
    int i = 1;
    int count = 0;
    while (i > 0) {
```

```
            i *= 2;
            count++;
        }
        cout << "Completed " << count << " iterations"<<endl;
        return 0;
    }
```

Your friend Rufus argues that this program has an infinite loop, and will therefore never end. Is he right or wrong? Justify your opinion with a persuasive explanation, but do not type in or run the program. Predict the output of the program.

11. For this class, you are required to use a Linux programming environment. Take this opportunity to set up such an environment. Use one of the options enumerated in the syllabus: Anubis, Vital, VirtualBox, WSL, or a proper Linux installation.

    Log in to your Linux environment. Then, submit a screenshot of your working Linux programming environment.

    The screenshot should simply show the Linux desktop. You don't need to do anything after you log in.

12. The course presumes a satisfactory knowledge of programming. In addition, we want to know that you are able to create, edit, and run programs in a Linux environment. Demonstrate that you can do programming under Linux by following these instructions. Your TA will help you if you get stuck.

    (a) Log in to your Linux environment: Anubis, Vital, VirtualBox, WSL, or a proper Linux installation.

    (b) Open a text editor on your Linux environment.

    - If you're using Anubis, simply select New File from the File menu to open a new text editing window.
    - If you're using Vital or VirtualBox, you should have a program called Text Editor or `gedit` available.
    - In any case, ask your TA if you can't find the right program.

    Save your new file as either `fizzbuzz.cpp` (for C++) or `fizzbuzz.py` (for Python). Make sure that you save the file in your *home directory*, which will typically be `/home/anubis` on Anubis, or `/home/ubuntu` on Vital.

    (c) Use the text editor to write a program in either Python or C++. Your program should have the following behavior:

    > The program will print out all the numbers from 1 to 100 in order. However, for numbers that are multiples of three, the program will print "Fizz" instead of the number. For numbers that are multiples of five, it will print "Buzz" instead of the number. And finally, for numbers which are multiples of both three and five, it will print "FizzBuzz".

    Everyone should be able to write this program. If you are not able to write this program correctly on the first try, without outside help, you may not meet the prerequisites for this course. In that case, please contact your professor to discuss your options.

    (d) After you've written the program, open a Terminal in your Linux environment.

    - If you're using Anubis, select New Terminal from the Terminal menu to open a command prompt window.
    - If you're using Vital or VirtualBox, you should have a program called Terminal available.
    - In any case, ask your TA if you can't find the right program.

    (e) Run your program, after compiling it if necessary.

- To run your Python program, type `python3 fizzbuzz.py` and press enter. Your program should run. If you see an error message, make sure that your program is saved is the right place.
- To compile your C++ program, type `g++ fizzbuzz.cpp -o fizzbuzz` and press enter. If compilation is successful, you will see no error message, and a new executable file named `fizzbuzz` will be created by the compiler.
  To run the program, now type `./fizzbuzz` and press enter. Your program should run. If you see an error message, make sure that your program is saved is the right place.
- In any case, ask your TA if you're having trouble.

(f) Take a screenshot of the Terminal showing the execution of your program. Submit this screenshot.