**Reading.** Read the following sections in the textbook.
- Ch 4: Section 4.2
- Ch 3: Section 3.1 and 3.2

**Part 1: Logic Gates.** Complete the following questions regarding logic gates, transistor circuits, and truth tables.

1. Give a truth table, draw a circuit diagram from logic (AND) gates, and draw the corresponding transistor diagram for the Boolean expression “\((a \& \& b) \& \& c\)” (in logic: \(a \land b \land c\)).

2. Give a truth table, draw a circuit diagram from logic (OR) gates, and draw the corresponding transistor diagram for the Boolean expression “\((a \| \| b) \| \| c\)” (in logic: \(a \lor b \lor c\)).

3. Give a truth table, draw a circuit diagram from logic (AND and OR) gates, and draw the corresponding transistor diagram for the Boolean expression “\((a \& \& b) \| (a \& \& c) \| (b \& \& c)\)” (in logic: \((a \land b) \lor (a \land c) \lor (b \land c)\)).

4. Draw a circuit diagram using only NAND gates that performs the operation “\(!a\)” (in logic: \(\neg a\)).

5. Draw a circuit diagram using only NAND gates that performs the operation “\(a \& \& b\)”.

6. Draw a circuit diagram using only NAND gates that performs the operation “\(a \| \| b\)”.

7. Draw a circuit diagram using only NAND gates that performs the operation “\(a \text{ NOR } b\)”.

8. Consider the following circuit diagram containing a “feedback loop” with inputs \(r\) and \(s\) and outputs \(q\) and \(\bar{q}\). Assume that \(r\) and \(s\) are typically “off” (low voltage), but one of them can be “pulsed” (turned “on” via raised voltage) for a short period of time (i.e., either \(r\) or \(s\) can be pulsed, but not both). After being pulsed, the input is turned off again (set back to low voltage). Assume the circuit is started with \(r\) being pulsed (and then immediately turned off). Thus, all NOR inputs are 0 (low) except for \(r\), which is 1 (high) when the circuit is started.

![Circuit Diagram]

(a) Describe what happens to the circuit as it is started. Assume \(r\) is turned off by the pulse before the “feedback” begins (i.e., \(r\) goes back to 0 before the second NOR gate output enters the first NOR gate). In your description, be sure to state what happens to \(q\) and \(\bar{q}\) as \(r\) goes from being pulsed to going back to 0.
(b) After some time, assume $s$ is pulsed. Describe what happens to $q$ and $\bar{q}$ (including once $s$ is turned off again after the pulse).

(c) Some time after $s$ was pulsed, assume $r$ is pulsed again. Describe what happens to $q$ and $\bar{q}$ (including once $r$ is turned off again after the pulse).

(d) What could such a circuit be used for?

Part 2: Getting Started with C. In this class you will become familiar with basic programming in C, which is very close to C++. Some of the major differences between the two includ C not having classes and using a different approach for input/output (e.g., `printf` instead of `cout`).

9. Copy the following C program into a file called `hw1_test1.c`.

```c
int main()
{
    int x = 7;
    int y = 2;
    int z = x * y;
    return 0;
}
```

Step 1: On either `ada` or the department’s Ubuntu virtual machine, follow the instructions in the textbook to: (a) compile the program (using the command `gcc -Og hw1_test1.c -o hw1_test1`); (b) run the program (using the command `./hw1_test1`); and (c) “disassemble” the compiled program (using the `objdump -d hw1_test1` command). Note that when you run the program in (b) nothing should be output. Locate the section of the assembly in the disassembled code for the `main` function and write it down (as part of your answer). Note you only need to include the assembly instructions (e.g., “`mov $0x0,%eax`”).

Step 2: The `-Og` flag in `gcc` performs (minimal) optimizations of the source program. To see this, follow the same process as above, but compile your code using the `-00` flag (full command: `gcc -O0 hw1_test1.c -o hw1_test1`). As above, locate the section of assembly in the disassembled code for the `main` function and write it down. Finally, try to map the three lines in your main program (the two assignments and the addition) to the disassembled assembly instructions. Note that `mov` is a basic assembly instruction, and `movl` is a variant that performs a 32-bit move instruction. Constant values are represented in hexadecimal (e.g., “`0x07`”) and prefixed with a `$` sign.

10. Do steps 1 and 2 above but for the following program. Save the program in a file named `hw2_test2.c`.

```c
int main()
{
    int x = 7;
    int y = 0;
```
```c
if (x % 2 == 0)
    y = 2;
else
    y = 1;
return 0;
}
```

**What to turn in.** Please submit your answers to the above questions in class on the due date. Note that you can either provide typed or handwritten answers. However, your answers must be legible and clearly marked. Be sure to include your name and the date on your work.