CSE 311: Foundations of Computing I

Homework 1 (due Friday, October 4th at 11:00 PM)

Directions: *Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. However, you may use results from lecture, the theorems handout, and previous homeworks without proof.*

1. I Can't Blab Such Blibber Blubber! (25 points)

Translate these English statements into logic, making the atomic propositions as small as possible.

(a) [5 Points] Only if the stadium is full and the crowd is not subdued will we win the game.

(b) [10 Points] Define a set of *at most three* atomic propositions. Then, use them to translate all of these sentences about my plan to buy a watch into logic:

i) If the watch is no more than $25, then I will buy it.

ii) Unless it is on sale, I will not buy the watch.

iii) I will buy the watch only if it is on sale and not more than $25.

(c) [10 Points] Define a set of *at most four* atomic propositions. Then, use them to translate all of these sentences about how a stack data structure can be used into logic:

i) You can push onto but not pop from the stack if it is empty.

ii) Only if the stack is full can you pop from but not push onto it.

iii) The stack is neither empty nor full unless you cannot pop from it or cannot push onto it.

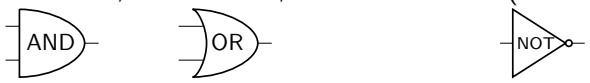
2. One, Two Three... How Many Fingers Do I See? (10 points)

Find a compound proposition involving the propositional variables p, q, and r that is true precisely when a majority of p, q, and r are true. Explain why your answer works.

3. Would You, Could You, With a Gate? (15 points)

Using only...

AND Gates, **OR** Gates, and Inverters (**NOT** Gates),



draw the diagram of a circuit with **three inputs** that computes the function $B(p, q, r)$, defined as follows: If both p and q are true, then $B(p, q, r) = \neg r$. If exactly one of p and q is true, then $B(p, q, r) = r$. Otherwise, we have $B(p, q, r) = \mathsf{F}$.

4. Aunt Annie's Alligator (20 points)

Define the "A" gate by the rule $A(r, s) := r \vee \neg s$.

Show how to implement the following gates using only A's. You are allowed to use the constants T and F and the inputs p and q. You are allowed to use inputs multiple times.

You must *justify your answers*.

(a) [5 Points] $\neg p$, using only **one** A gate

(b) [5 Points] $p \vee q$, using at most **two** A gates

(c) [5 Points] $\neg(p \wedge q)$, using at most **two** A gates

(d) [5 Points] $p \wedge q$, using at most **three** A gates

5. Thing One and Thing Two (20 points)

Use truth assignments (i.e., an assignment of a truth value to each variable) to demonstrate that the two propositions in each part are not logically equivalent. Explain why your assignment demonstrates this.

(a) [5 Points] $p \vee (p \wedge q)$ vs. $q \vee (p \wedge q)$

(b) [5 Points] $\neg(p \oplus q)$ vs. $\neg p \oplus \neg q$

(c) [5 Points] $p \to (q \to r)$ vs. $(p \to q) \to r$

(d) [5 Points] $(p \to q) \to (q \to p)$ vs. $(q \to p) \to (p \to q)$

6. The Curious Case of The Lying TAs (10 points)

A new UW CSE student wandered around the Paul Allen building on their first day in the major. They found (as many do) that there is a secret room in its basements. On the door of this secret room is a sign that says:

```
All ye who enter, beware!  Every inhabitant of this room is either a TA who always
 lies or a student who always tells the truth!
```

(a) [5 Points] The CSE student walked into the room, and two inhabitants walked up to the student. One of them said "at least one of us is a TA." Determine (with justification) all the possibilities for each of the two inhabitants.

(b) [5 Points] Three inhabitants walk up to the CSE student and surround the UW CSE student. One of them says "every TA in this circle has a TA to his immediate right." Determine (with justification) all the possibilities for each of the three inhabitants.

7. EXTRA CREDIT: XNORing (0 points)

Imagine a computer with a fixed amount of memory. We give names, $R_1, R_2, R_3, \ldots$, to each of the locations where we can store data and call these "registers." The machine can execute instructions, each of which reads the values from some register(s), applies some operation to those values to calculate a new value, and then stores the result in another register. For example, the instruction $R_4 := \texttt{AND}(R_1, R_2)$ would read the values stored in R_1 and R_2, compute the logical and of those values, and store the result in register R_4.

We can perform more complex computations by using a sequence of instructions. For example, if we start with register R_1 containing the value of the proposition p and R_2 containing the value of the proposition q, then the following instructions:

1. $R_3 := \texttt{NOT}(R_1)$
2. $R_4 := \texttt{AND}(R_1, R_2)$
3. $R_4 := \texttt{OR}(R_3, R_4)$

would leave R_4 containing the value of the expression $\neg p \vee (p \wedge q)$. Note that this last instruction reads from R_4 and also stores the result into R_4. This is allowed.

Now, assuming p is stored in register R_1 and q is stored in register R_2, give a sequence of instructions that

- only uses the `XNOR` operation (no `AND`, `OR`, etc.),
- only uses registers R_1 and R_2 (no extra space), and
- ends with q stored in R_1 and p stored in R_2 (i.e., with the original values in R_1 and R_2 swapped).