CSE 311: Foundations of Computing I — Winter 2026

Problem Set 1

Due: Friday, Jan 16th by **11:00pm**

Instructions

Write up carefully argued solutions to the following problems. Each solution should be clear enough that it can explain (to someone who does not already understand the answer) why it works. However, you may use results from lecture, the reference sheets, and previous homework without proof.

Collaboration policy. You are required to submit your own solutions. You are allowed to discuss the homework with other students. However, the **write up** must clearly be your own, and moreover, you must be able to explain your solution at any time. We reserve ourselves the right to ask you to explain your work at any time in the course of this class.

Late policy. You have a total of **three** late days during the quarter, but you can only use one late day on any one problem set. Please plan ahead. The final problem set will not be accepted late.

Solutions submission. Submit your solution via Gradescope. In particular:

- Each numbered task should be solved on its own page (or pages). Do not write your name on the individual pages. (Gradescope will handle that.)
- When you upload your pages, make sure each one is **properly rotated**. If not, you can use the Gradescope controls to turn them to the proper orientation.
- Follow the Gradescope prompt to **link tasks to pages**.
- You are not required to typeset your solution, but your submission must be **legible**. It is your responsibility to make sure solutions are readable — we will *not* grade unreadable write-ups.

Task 1 – Good to the Last Prop [8 pts]

Consider this sentence: Andre goes out with friends if it's a Friday or if Andre doesn't have both homework and an exam on the next day.

a) Define a set of *four* atomic propositions. Then, use them to translate the sentence above into propositional logic.

b) Take the contrapositive of the logical statement from part (a). Then, rewrite it so that all $\neg$ symbols are next to atomic propositions, by applying De Morgan's law and Double Negation as necessary.

c) Translate the sentence from part (b) back to English.

d) Assuming translation between English and propositional logic is unambiguous, does your English sentence from part (c) have the same truth value as the original English sentence above? Why or why not?

Task 2 – With a Fine-Truth Comb [6 pts]

For each of the following pairs of propositions, use truth tables to determine whether they are equivalent.

Include the full truth table and state whether they are equivalent. (In principle, only one row is needed to show non-equivalence, but please turn in the entire table so that we can give partial credit in case of errors.) Your truth table must include columns for all subformulas.

a) $P \oplus Q$ vs. $\neg P \oplus (P \wedge Q)$

b) $(P \rightarrow Q) \rightarrow R$ vs. $(P \vee R) \wedge (Q \rightarrow R)$

Task 3 – Too Cool For Rule [10 pts]

Prove the following assertions using a sequence of logical equivalences such as Absorption, Associativity, Commutativity, Contrapositive, De Morgan, Distributivity, Double Negation, Idempotency, Identity, Law of Implication, and Negation.

Hints: For equivalences where one side is much longer than the other, a good heuristic is to start with the longer side and try to apply the rules that will shorten it. In some cases, it may work better to work to shorten both sides to the same expression and then combine those two sequences into one.

a) $(P \vee \neg Q) \wedge (P \wedge \neg Q) \equiv P \wedge \neg Q$ (*Hint*: It is **not** necessary to use Distributivity here.)

b) [1] $(\neg Q \rightarrow R) \wedge (Q \rightarrow R) \equiv R$

c) $(P \vee R \rightarrow Q) \rightarrow R \equiv (\neg P \rightarrow R) \wedge (Q \rightarrow R)$

[1] This corresponds to what is commonly known as proof by "simple cases"; if one can prove that R holds when Q is true and also when Q is false, then R must always hold.

Task 4 – Fizz, and Only Fizz [12 pts]

Shane practices a personal ritual every day to decide whether he will drink a ginger ale. He flips a coin 4 times, and assigns values of 1 (heads) or 0 (tails) to variables a, b, c, and d based on the results of the flips, in that order. Shane then applies a secret boolean function G to those variables, and drinks a ginger ale if and only if $G(a, b, c, d) = 1$.

Ilya wants to figure out the secret function G. After extensive observation, he has recorded a truth table for G, shown below. Help Ilya reconstruct the function!

a	b	c	d	$G(a, b, c, d)$
1	1	1	1	1
1	1	1	0	1
1	1	0	1	0
1	1	0	0	0
1	0	1	1	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	0
0	1	1	1	1
0	1	1	0	1
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1
0	0	1	0	1
0	0	0	1	0
0	0	0	0	0

a) Write a Boolean algebra expression for G in canonical DNF form [2].

b) Use equivalences of Propositional Logic to simplify your expression from (a) down to an expression that includes *only 3 gates* (each of which is either AND, OR, or NOT).

You should format your work like an equivalence chain with one expression per line and with the name of the identity applied to produce that line written next to it. However, since we are using Boolean algebra notation, which does not include unnecessary parentheses, you do not need to include lines that apply Associativity, Commutativity, or Identity (you may still cite them for clarity if desired).

c) Write a truth table for your simplified expression from part (b) and confirm that it matches the one used to define G originally. As always, be sure to include all subexpressions as their own columns.

d) Draw your simplified expression from part (b) as a circuit.

[2]With circuits, these are usually called sum-of-products

Task 5 – The Odd One Out [14 pts]

The following parts use the predicates Odd, Prime, and IsTwo, defined as follows:

$$\begin{aligned} \mathsf{Odd}(x) &:= \exists k\,(x = 2k+1) \\ \mathsf{Prime}(x) &:= \neg(\exists j \exists k((jk = x) \land (j \neq 1) \land (j \neq x))) \land (x \geqslant 2) \\ \mathsf{IsTwo}(x) &:= (x = 2) \end{aligned}$$

Note: Prime will be true for prime numbers $\{2, 3, 5, 7, 11, \ldots\}$ and false for all other inputs (a natural is prime if it has no divisors other than 1 and itself). For this problem, you *do not* need to make your translation sound natural (literal is fine).

a) Let the domain of discourse be all non-negative integers (the naturals: $0, 1, 2, \ldots$).

i) Translate the proposition

$$\forall x\,(\mathsf{Prime}(x) \rightarrow \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

ii) Translate the proposition

$$\forall x\,((\mathsf{Prime}(x) \land \neg\,\mathsf{IsTwo}(x)) \rightarrow \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

iii) Translate the proposition

$$(\forall x\,(\mathsf{Prime}(x) \land \neg\,\mathsf{IsTwo}(x))) \rightarrow (\forall x\ \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

b) Let the domain of discourse be all non-negative *even* integers (the even naturals: $0, 2, 4, 6, \ldots$).

i) Translate the proposition

$$\forall x\,(\mathsf{Prime}(x) \rightarrow \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

ii) Translate the proposition

$$\forall x\,((\mathsf{Prime}(x) \land \neg\,\mathsf{IsTwo}(x)) \rightarrow \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

iii) Translate the proposition

$$(\forall x\,(\mathsf{Prime}(x) \land \neg\,\mathsf{IsTwo}(x))) \to (\forall x\ \mathsf{Odd}(x))$$

directly into English. Then, state whether the proposition is true or false (no justification needed).

c) Let P and Q be predicates, and fix a domain of discourse. Given that $\forall x(P(x) \to Q(x))$ is true, is $\forall x\,(P(x)) \to \forall x\,(Q(x))$ always true? Justify your answer with 1-2 sentences.

Task 6 – Extra Credit: XNORing

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 some 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 A and R_2 containing the value of the proposition B, 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 $\neg A \vee (A \wedge B)$. Note that this last instruction reads from R_4 and also stores the result into R_4. This is allowed.

Now, assuming A is stored in register R_1 and B 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 B stored in R_1 and A stored in R_2 (i.e., with the original values in R_1 and R_2 swapped).