Guideline

Homework for Week 01

Please submit your solutions as a single PDF on Brightspace.

Multiple submissions are allowed before the due time; the last submission will be graded.

Grading Criteria

- Grading will be *highly strict*, with minimal tolerance for mistakes or misconduct.
- Factual grading errors will be corrected, but partial grading decisions will not be negotiable.
- If the solution appears to be plagiarized or AI-generated, the issue will be reported to the instructor.
- Most questions here involve constructing truth tables and drawing a conclusion. If the conclusion is incorrect, 75% of the total points will be deducted. If the truth table is incorrect, 1 point to 75% of the total points may be deducted, depending on the severity of the errors. The total deduction cannot exceed the full score.

Exercise 0 (5 points)

Indicate whether each statement regarding our class policies is True (T) or False (F).

1. Late homework is totally acceptable if the student is busy.
2. If a student strolls into class late after the attendance checks have been done, they will still get full attendance credit.
3. Actively asking questions during class, or providing constructive feedbacks, can positively impact the student’s final grade.
4. Copying answers directly from AI tools is acceptable because AI is cool.
5. A student's class absence can be excused if they submit a document showing they simply visited a doctor.

Exercise 1 (20 points)

Check if the two statements below are equivalent using the truth table technique. Your conclusion should be either "equivalent" or "not equivalent."

- (p -> q) /\ (q -> r) /\ (r -> p)
- p /\ q /\ r

Exercise 2 (20 points)

Consider the six statement forms (a-f):

- (a) p -> q
- (b) q -> p
- (c) ~p \/ q
- (d) ~q \/ p
- (e) ~q -> ~p
- (f) ~p -> ~q

1. Find all statement forms that are equivalent to (a), except (a) itself.
2. Find all statement forms that are equivalent to (b), except (b) itself.

For this exercise, constructing truth tables is needed.

Exercise 3 (20 points)

Consider the proposition ~P /\ (Q -> P). What can you conclude about P and Q if you know the statement is true? For this exercise, an explanation is not needed. Just show your results using a truth table.

Exercise 4 (15 points)

A tautology and a contradiction are terms used in logic to describe specific types of propositions.

- A *tautology* is a proposition that is always true, regardless of the truth values of its constituent parts. In other words, a proposition is a tautology if it is true in every row of a truth table. For example, $p \lor {\sim}p$. An example in plain language: "It will either rain today or it won't."

- A *contradiction* is a proposition that is always false, regardless of the truth values of its constituent parts. In other words, a proposition is a contradiction if it is false in every row of a truth table. For example, $p \land {\sim}p$. An example in plain language: "I will finish the homework today and I will not."

For each statement form below, determine if it is a tautology, contradiction, or neither. For this exercise, an explanation is not needed. Your answer should be "tautology," "contradiction," or "neither."

1. $({\sim}p \lor q) \lor (p \land {\sim}q)$
2. $(p \land {\sim}q) \land ({\sim}p \lor q)$
3. $(p \land q) \lor ({\sim}p \lor (p \land {\sim}q))$

Exercise 5 (20 points)

Verifying Simple Code with Hoare Logic

Hoare Logic is a formal system for reasoning about the correctness of computer programs. Tony Hoare introduced it in 1969, which contributed to his Turing Award. Hoare Logic helps in verifying that a program does what it is supposed to do. Interested students should read this: https://amturing.acm.org/award_winners/hoare_4622167.cfm

Hoare Logic revolves around the concept of a Hoare Triple, which has the following format:

```
{P} Code {Q}
```

- P is the precondition. It's a proposition about the variables in your program that is true before the code runs.
- Q is the postcondition. It's a proposition that you expect to be true after the code runs.
- Code is the program or part of the program that you're analyzing.

Examples of correct Hoare Triples:

- {x = 1} `y := x + 1` {y = 2 ⋀ x = 1}
- {x = 1} `y := x + 1` {y = 2}
- {x = 0} `while (x<=10) {x = x + 1}` {x = 11}

Note: In the above examples, "y = 2" is weaker than "y = 2 ⋀ x = 1," but it remains correct after `y := x + 1` is executed with the precondition "x = 1."

Examples of incorrect Hoare Triples:

- {x = 1} `y := x + 1` {y - x = 2}
- {x > 0} `while (x<=10) {x = x + 1}` {x = 11}

Note: The second example is incorrect because it claims that if "x > 0," then "x" will equal 11 at the end of the loop. This is flawed for two reasons: (1) If "x" starts as 0.5, then at the end of the loop, "x" will be 10.5, not 11. (2) If "x" starts as a value greater than 10, such as 42, then the loop will terminate without changing the value of "x" at all.

Questions: Determine whether the following Hoare Triple is correct or not. For this exercise, an explanation is not needed. Your answers should be "correct" or "not correct."

1. {x = 3 ⋀ y = 2} `Code` {x = 2 ⋀ y = 3}, where "Code" refers to the following:

```
x := x + y
y := x − y
x := x − y
```

2. {x > 0} `while (x<=10) {x = x + 1}` {x > 10}. Note this one looks similar to, but different from, a piece of code we discussed in class.