# Guideline

Homework for week 01. Due: 23h59 on Thursday, 2023-09-07.

Please submit your solutions in a single PDF on Brightspace.

If you're writing by hand, please ensure your handwriting is legible.

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

# Exercise 1. (points = 10)

Check if the two statement forms below are logically equivalent. Explanation is needed based on the truth table.

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

# Exercise 2 (points = 10)

Check if the two statement forms below are equivalent. Explanation is needed based on the truth table.

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

# Exercise 3 (points = 30)

For this exercise, explanation is needed based on the truth table.

Consider 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.

# Exercise 4 (points = 5)

Consider the proposition ~P /\ (Q -> P). What can you conclude about P and Q if you know the statement is true? Explanation is needed based on the truth table.

# Exercise 5 (points = 15)

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 \/ ~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 /\ ~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. Explanation is needed based on the truth tables.

1. (~p \/ q) \/ (p /\ ~q)
2. (p /\ ~q) /\ (~p \/ q)
3. (p /\ q) \/ (~p \/ (p /\ ~q))

# Exercise 6. (points = 5)

Check if the two statement forms below are logically equivalent and explain with truth tables.

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

# Exercise 7. (points = 15)

## Smart Home Security System

Consider a house that is equipped with a smart security system. The system uses a combination of a facial recognition camera and a door sensor to control the main entrance lock.

- The facial recognition camera detects if a family member's face is recognized.

- The door sensor detects if the door is closed.

The main entrance lock will unlock if: A family member's face is recognized *AND* the door is closed.

Let $Face$ represent the facial recognition result, where $Face$ = T if a family member's face is recognized and $Face$ = F if not.

Let $Door$ represent the door sensor where $Door$ = T if the door is open and $Door$ = F if the door is closed.

Let $Lock$ represent the main entrance lock, where $Lock$ = T if the lock is locked and $Lock$ = F if the lock is not locked.

Questions:

1. Construct a truth table based on the above conditions and symbols.

   Hint: This truth table should have three columns of $Face, $Door$, and $Lock$.

2. Analyze the table to determine under what conditions the door will unlock.

3. As an extension, think of real-world scenarios where it might be beneficial to add more conditions (e.g., time of day, alarm set status) and describe how they might affect the logic of the security system. Explain in plain words, and maybe update your truth table.

# Exercise 8 (points = 10)

### 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 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, which you're analyzing.

Examples of correct Hoare Triple

- {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: Above y = 2 is weaker than y = 2 ⋀ x = 1, but it remains to be correct after `y := x + 1` is executed from precondition x = 1.

Examples of wrong Hoare Triple:

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

Note: The second is incorrect because it states that: If x > 0, 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 is 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:

1. $\{x = 3 \wedge y = 2\}$ `Code` $\{ x = 2 \wedge y = 3\}$, where "Code" refers to the following

```
x := x + y
y := x - y
x := x - y
```

2. $\{x > 0\}$ `while (x<=10) {x = x + 1}` $\{x > 10\}$