

Final Project

Out: 11/20/2025 ; Due: 12/10/2025)

November 20, 2025

Submission instructions:

Please submit the project on D2L in time. The **deadline is strict!** Submit a PDF report that includes **all** the names of the members of the group and describes what you did in full details. Please copy/paste the code you produced into the report (no screenshots). Just one person of the group should submit the report!

Project instructions:

This project requires producing small C code snippets that both *satisfy* and *falsify* a given Frama-C specification. Your task is to prompt the LLM so that it generates the body of the function for each given specification.

For each part (P1, P2, P3, P4), you must provide two code samples:

- one implementation that *satisfies* the specification, meaning that all verification goals are validated by Frama-C,
- one implementation that *falsifies* the specification, meaning that at least one verification goal is not validated.

You will be given a Frama-C specification that must appear at the top of each code file. The LLM must generate the C code that fills in the function body so that the resulting program satisfies or falsifies the specification.

A satisfying implementation may require adding loop invariants or auxiliary assertions, e.g., loop variants/terminating functions for total correctness, for Frama-C to validate some of the goals.

You are not allowed to instruct the LLM to violate post-conditions intentionally. The prompts for both implementations (satisfying and falsifying) should be consistent.

For each of the parts given below, the report should provide (in order):

- A short (one-paragraph) discussion of what you did and why.
- The LLM model used to generate the code.
- If used, the temperature (or other generation settings).
- The details of the interaction between you and the LLM:
 - The exact prompt sequence provided to the LLM to reach the final result together with the corresponding replies by the LLM and their exact time stamps.

- The timestamp indicating when the final code was generated.
- The complete and final code produced by the LLM.
- The Frama-C reports obtained using:

`frama-c -wp code_by_LLM.c -then -report`

Note: If you tried multiple times and could not fully succeed, it is fine to submit a sub-optimal solution for partial credit. You can still earn a good grade!

1. **P1. (15 pts)** In this part, you must return code that swaps two elements of an array using arithmetic or bitwise operations, according to the following Hoare triple. Your prompt to the LLM should instruct it to implement code satisfying:

$$\left\{ \forall i \in [0, n). a[i] = V[i] \wedge 0 \leq k < n \wedge 0 \leq j < n \wedge k \neq j \right\}$$

`code_by_LLM`

$$\left\{ a[k] = V[j] \wedge a[j] = V[k] \wedge \forall i \in [0, n). i \neq k \wedge i \neq j \Rightarrow a[i] = V[i] \right\}$$

2. **P2. (25 pts)** In this part, you must return code that determines whether an array is *strictly descending*, according to the following Hoare triple. Your prompt to the LLM should instruct it to implement code satisfying:

$$\{ \forall k \in [0, n). a[k] = V[k] \}$$

`code_by_LLM`

$$\begin{cases} (desc = 1) \Rightarrow (\forall i, j. 0 \leq i < j < n \wedge V[i] > V[j]) \\ (desc = 0) \Rightarrow (\exists i, j. 0 \leq i < j < n \wedge V[i] \leq V[j]) \end{cases}$$

3. **P3. (30 pts)**

In this part, you must return an iterative code that rotates an array to the left by 1 index, according to the following Hoare triple. Your prompt to the LLM should instruct it to implement code satisfying:

$$\{ \forall i \in [0, n). a[i] = V[i] \}$$

`code_by_LLM`

$$\{ \forall i \in [0, n - 1). a[i] = V[i + 1] \wedge a[n - 1] = V[0] \}$$

4. **P4. (30 pts)** In this part, you must return code that transforms each array element according to the condition

$$V[k] \leq 0 \Rightarrow a[k] = 0, \quad V[k] > 0 \Rightarrow a[k] = k.$$

Your prompt to the LLM should instruct it to implement code satisfying:

$$\{\forall k \in [0, n). a[k] = V[k]\}$$

`code_by_LLM`

$$\{\forall k \in [0, n). (V[k] \leq 0 \Rightarrow a[k] = 0) \wedge (V[k] > 0 \Rightarrow a[k] = k)\}$$