Question 8 (Bonus Question)

Model used: Llam 3.1 70B parameter

The LLaMA 3.1 70B model is a transformer-based language model with 32 layers, 8,192 hidden size, and approximately 70 billion parameters. It's capable of generating code, but its proficiency depends on the specific task and programming language. Here's a brief assessment:

Code generation capabilities:

Good at: Generating boilerplate code, simple algorithms, and data structures in languages like Python, Java, and JavaScript. Fair at: Writing moderately complex code, such as API integrations, web scraping, and data analysis. Limited at: Creating highly complex or optimized code, like low-level system programming, high-performance computing, or expert-level algorithms.

Here's what the process looked like:

Step 1: I began by crafting a simple prompt template. It provided the question as context and asked the LLM to assist in writing a Prolog solution.

Step 2: The initial code sample the LLM generated wasn't recursive. Instead, it used an accumulator pattern, where attributes were passed along and the final case equated them to the expected solution.

Step 3: I then prompted it to switch gears and generate a recursion-based approach, starting with just the base case.

Step 4: Once I had the base case, I asked it to build out the rest of the recursive logic.

Step 5: The result was a rather complex code. I had to guide the LLM by specifying which constructs to avoid, like the disjunction (";"), cut ("!"), and if-then ("->") operators.

Step 6: Eventually, it delivered a working solution. However, the code was still more intricate than necessary and would benefit from human refinement to simplify it further.