Matematická logika

Úloha ku skúške

1 Introduction

In data processing, a *stable partition* algorithm separates the elements of a sequence into two groups based on a boolean predicate while preserving the relative order of the elements.

Your Core Task

Implement in Lean 4 a function `split_parity` that computes the stable partition of a given list of natural numbers into two lists: one containing only **even** numbers and another containing only **odd** numbers. Formally prove that the implementation is correct.
Function Signature:

```
def split_parity : List Nat → List Nat × List Nat
| [] => ([], [])
| (x :: xs) =>
  -- Your implementation here
```

2 Verification Tasks

Complete the following proofs. You may define helper lemmas if needed.

2.1 Task 1: Length Conservation

```
theorem split_parity_length (xs : List Nat) :
  (split_parity xs).1.length + (split_parity xs).2.length = xs.length := by
  sorry
```

2.2 Task 2: Correctness of Parity

```
theorem split_parity_all_even (xs : List Nat) :
  ∀ x ∈ (split_parity xs).1, Even x := by
  sorry

theorem split_parity_all_odd (xs : List Nat) :
  ∀ x ∈ (split_parity xs).2, Odd x := by
  sorry
```

2.3 Task 3: Conservation of Elements

```
theorem split_parity_mem (xs : List Nat) (x : Nat) :
  x ∈ xs ↔ x ∈ (split_parity xs).1 ∨ x ∈ (split_parity xs).2 := by
  sorry
```

3 Bonus Task: Stability

Prove that the partition is *stable* (the original order is preserved). You should formulate this property yourself, perhaps using the `List.Sublist` relation from Mathlib.

4 Rules and Expectations

- **Experimental Nature:** This is a challenging task. You do not need to complete every theorem to receive a high grade. Well-documented attempts and partial proofs are encouraged.
- **Allowed Resources:** You may use Mathlib, online documentation, and textbooks. You may use AI for general questions or syntax help if you get stuck, but you **must not** allow an AI to generate the full proofs (exception is the bonus task).
- **Oral Exam:** You will be asked to explain your logic line-by-line during an oral defense.

5 Technical Reference

Use the following imports for Mathlib parity definitions:

```
import Mathlib.Data.List.Basic
import Mathlib.Data.Nat.Parity
```

Note the difference between **computation** (using `n % 2 == 0`) and **propositions** (using `Even n`). You will need to bridge these using theorems like `Nat.even_iff`.