Report: Exercise 7 - Variant 4 (Words to Numbers)

Student Name: Aayush rautela & Cagla Kuleasan

1. Logic Flow of Our Solution

Our goal was to convert English number words (up to "one thousand") into digits. We approached this by:

1. **Preprocessing Input:** We convert the input string to lowercase and split it into a list of word atoms (e.g., ['one', 'hundred', 'and', 'five']).

2. **Pattern Matching and Parsing:** Our solution uses Prolog rules (parse_number_from_list/2) to match patterns in this word list. These rules cover structures like "one thousand," hundreds with "and" (e.g., "two hundred and thirty four"), hundreds alone, tens with units, tens alone, teens, and units. The order of these rules is critical: more specific (longer) patterns must precede general ones for correct parsing.

3. **Word-to-Value Mapping:** We use helper predicates (unit_word/2, teen_word/2, tens_word/2) to map individual number words to their integer values.

4. **Calculation:** Upon matching a pattern, we retrieve numerical values using helper predicates and combine them arithmetically to get the final number.

5. **Output:** The main predicate to_num/2 provides the final numerical result.

Our logic is a top-down parsing approach: tokenizing the input, then matching tokens against grammatical patterns for number words, and finally performing calculations.

2. Main Components of Our Code

Our code has three main parts:

1. **Word-to-Value Facts (unit_word/2, teen_word/2, tens_word/2):**
    - These facts map number word atoms to integer values (e.g., unit_word(one, 1)).
    - Purpose: Provide basic numerical building blocks.
2. **Main Conversion Predicate (to_num/2):**
    - to_num(WordString, Number) is the public interface.
    - It preprocesses the input string and calls parse_number_from_list/2 for parsing and calculation.
    - Purpose:Make the conversion process easy to manage through a simple interface.
3. **Parsing Rules (parse_number_from_list/2):**
    - A series of clauses, each defining a word pattern and its conversion to a number (e.g., parse_number_from_list([one, thousand], 1000).).
    - These rules use helper facts for word values and is/2 for arithmetic. Their order is crucial.
    - Purpose: Implement the core logic for recognizing and calculating number word structures.

3. Results

```
?- to_num("one hundred and ninty five", N).
false.

?- to_num("one hundred and ninety five", N).
N = 195 .

?- to_num("eight", N).
N = 8.

?-  to_num("thirteen", N).
N = 13 .

?- to_num("seventy", N).
N = 70 .

?- to_num("sixty five", N).
N = 65 .

?- to_num("four hundred and two", N).
N = 402 .

?- to_num("four hundred and two", N).
N = 402 .

?- to_num("one thousand", N).
N = 1000
```

4. Challenges We Faced

Absolutely no experience in dealing with prolog. Took a lot of time to fix syntax errors.