Laboratory 7
Variant 3
Group 3

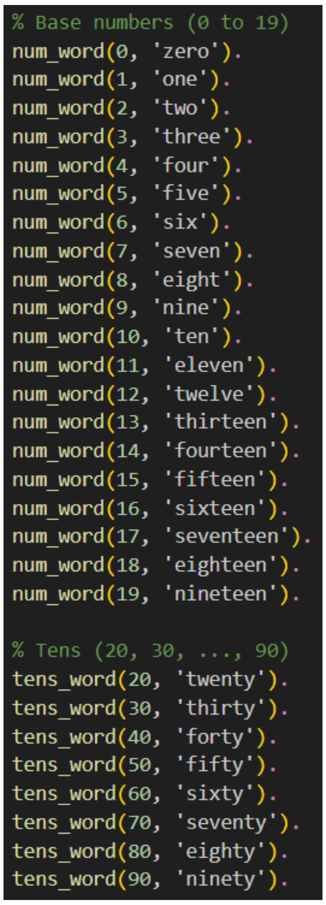
By Krzysztof Kotowski and Juan Manuel Aristizabal Henao

Introduction

In this laboratory task, we were tasked to create a Prolog script that would solve Variant number 3, which is about converting an input number N to English words where N is less or equal to 1000.

Implementation

The code starts by defining some facts that are related to the direct conversion from known numbers to their English word equivalents. This is done for numbers between 0 and 19, and for the numbers equivalent to tens (e.g., 20, 30, 40, 50, 60, 70, 80, 90).

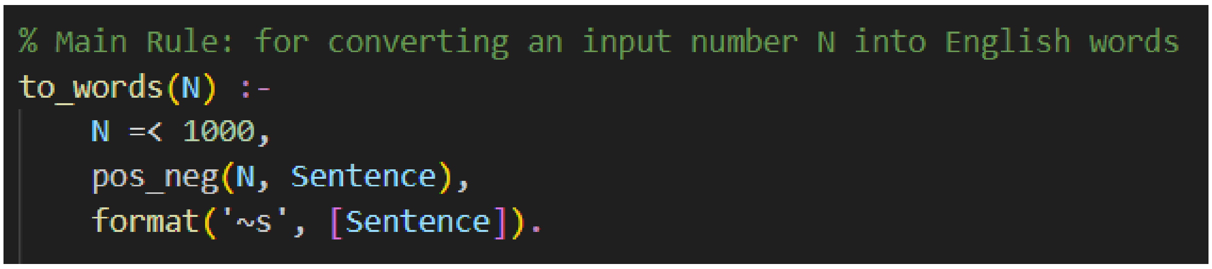


```prolog
% Base numbers (0 to 19)
num_word(0, 'zero').
num_word(1, 'one').
num_word(2, 'two').
num_word(3, 'three').
num_word(4, 'four').
num_word(5, 'five').
num_word(6, 'six').
num_word(7, 'seven').
num_word(8, 'eight').
num_word(9, 'nine').
num_word(10, 'ten').
num_word(11, 'eleven').
num_word(12, 'twelve').
num_word(13, 'thirteen').
num_word(14, 'fourteen').
num_word(15, 'fifteen').
num_word(16, 'sixteen').
num_word(17, 'seventeen').
num_word(18, 'eighteen').
num_word(19, 'nineteen').

% Tens (20, 30, ..., 90)
tens_word(20, 'twenty').
tens_word(30, 'thirty').
tens_word(40, 'forty').
tens_word(50, 'fifty').
tens_word(60, 'sixty').
tens_word(70, 'seventy').
tens_word(80, 'eighty').
tens_word(90, 'ninety').
```

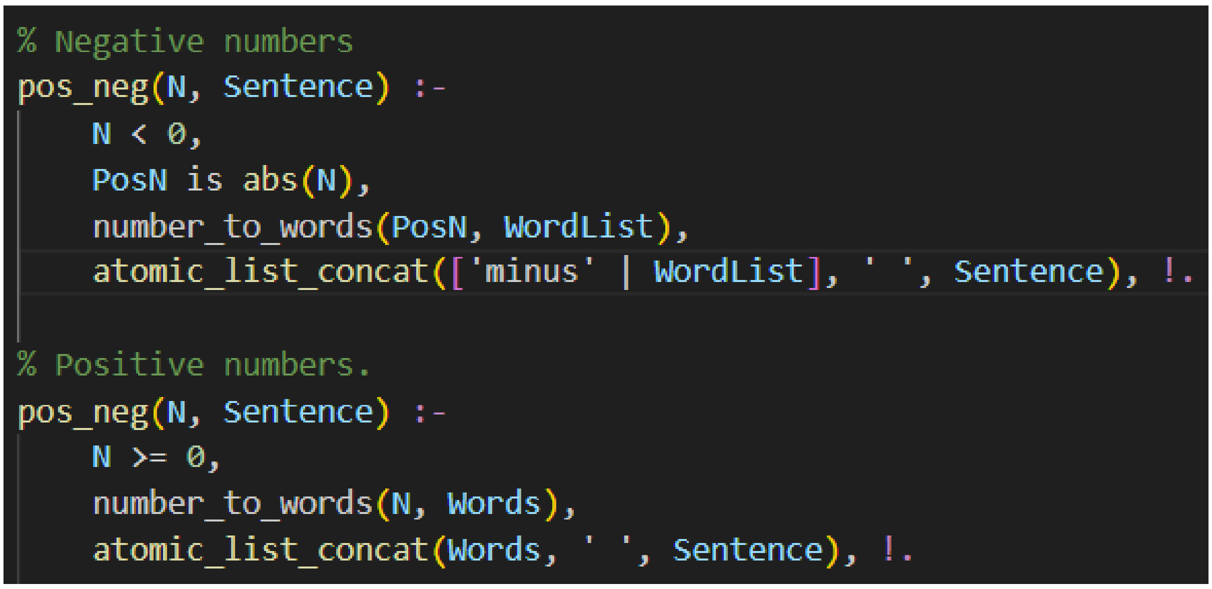
Then the rules are declared starting with the main rule (e.g. to_words(N)), which when called will convert the specified number N into English words equivalent. The flow of this rule is the following:

1. it is checked that the N parameter is less than or equal to 1000.
2. Then the rule for creating a string out of the N parameter is called (e.g., pos_neg(N))
3. The returned sentence is formatted to be printed as a string.



```
% Main Rule: for converting an input number N into English words
to_words(N) :-
    N =< 1000,
    pos_neg(N, Sentence),
    format('~s', [Sentence]).
```

The rule pos_neg(N, Sentence) is responsible for identifying whether the number is positive (N >= 0) or negative (N < 0). In case the number is negative, a 'minus' will be prepended to the Sentence, and the absolute value of N will be used to find its English words equivalent.

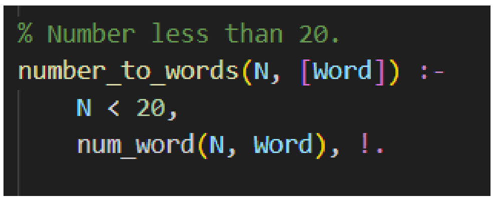


```
% Negative numbers
pos_neg(N, Sentence) :-
    N < 0,
    PosN is abs(N),
    number_to_words(PosN, WordList),
    atomic_list_concat(['minus' | WordList], ' ', Sentence), !.

% Positive numbers.
pos_neg(N, Sentence) :-
    N >= 0,
    number_to_words(N, Words),
    atomic_list_concat(Words, ' ', Sentence), !.
```

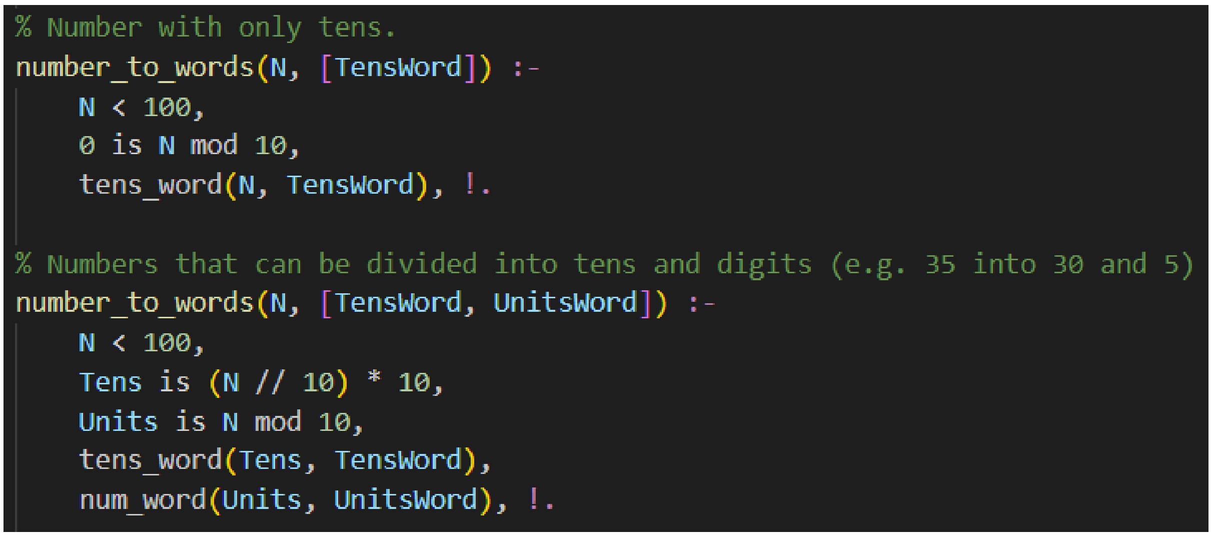
The rule number_to_words(N, WordList) is used to identify the different parts of the given number N, and return a list containing all their English word equivalents. The parts identified will be either Units (numbers less than 20 or digits), Tens (will be either single tens as defined in the facts before, or a number containing both tens and units) or Hundreds (will be either single hundred number or a hundred number containing some remainder.)

For identifying units, the following is the flow of the rule:



```
% Number less than 20.
number_to_words(N, [Word]) :-
    N < 20,
    num_word(N, Word), !.
```

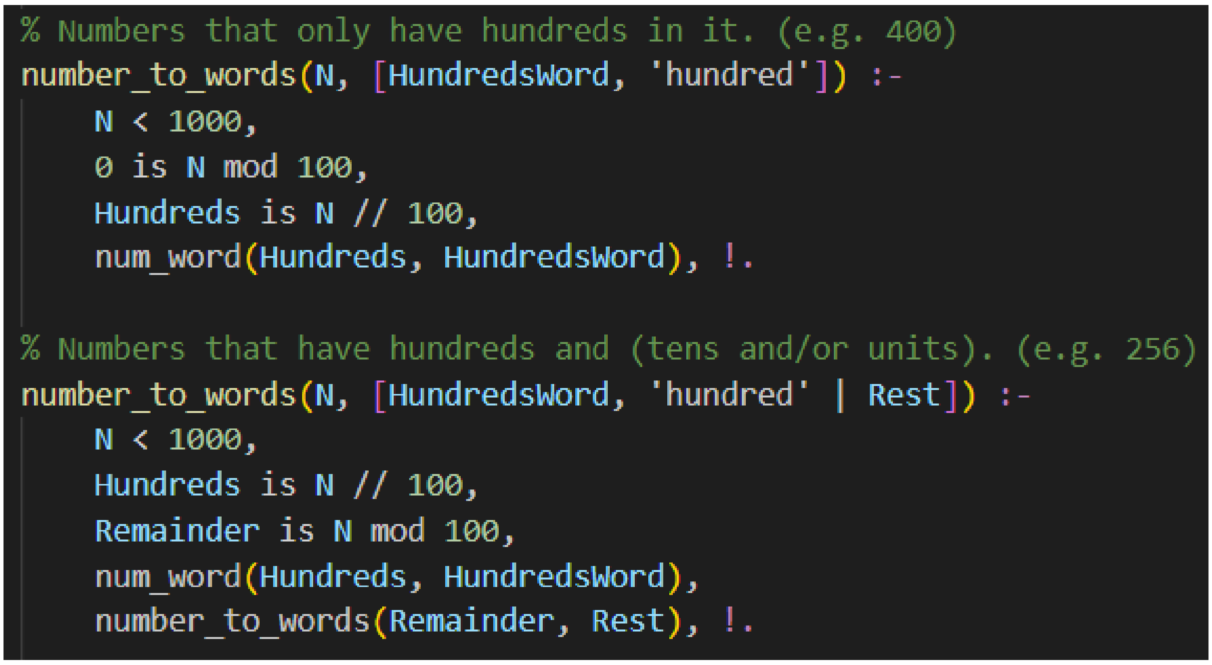
For identifying tens, the rule first checks if N has no remainder when dividing it by 10; if so, the first flow is used, otherwise the second flow is used. On the second flow, the number N is divided into both its tens and units, and both are handled separately.



```prolog
% Number with only tens.
number_to_words(N, [TensWord]) :-
    N < 100,
    0 is N mod 10,
    tens_word(N, TensWord), !.

% Numbers that can be divided into tens and digits (e.g. 35 into 30 and 5)
number_to_words(N, [TensWord, UnitsWord]) :-
    N < 100,
    Tens is (N // 10) * 10,
    Units is N mod 10,
    tens_word(Tens, TensWord),
    num_word(Units, UnitsWord), !.
```

For identifying hundreds, the rule first checks if the number N has no remainder when dividing it by 100; if so, the equivalent English word for the most significant digit is searched for in the declared facts and the word 'hundred' is appended. In the other hand, if there is a remainder, the second flow is used, where the most significant digit and the remainder of the operation are separated, for the former, the English word equivalent is looked for, and for the latter, it is treated as a number with possible tens and/or units parts.



```prolog
% Numbers that only have hundreds in it. (e.g. 400)
number_to_words(N, [HundredsWord, 'hundred']) :-
    N < 1000,
    0 is N mod 100,
    Hundreds is N // 100,
    num_word(Hundreds, HundredsWord), !.

% Numbers that have hundreds and (tens and/or units). (e.g. 256)
number_to_words(N, [HundredsWord, 'hundred' | Rest]) :-
    N < 1000,
    Hundreds is N // 100,
    Remainder is N mod 100,
    num_word(Hundreds, HundredsWord),
    number_to_words(Remainder, Rest), !.
```

Finally, if the number N is equivalent to 1000, a fact is used which returns a list containing both 'one' and 'thousand'.

Challenges

The solution for this task may not have been the most efficient, but from the tutorials taken, at least this code is guaranteed to work for inputs of N between (-1000, 1000). Hence, it is assumed that the inputs might be negative, even though it was not specified as such.

The code was only tested using the online environment. Hence, for local testing, we are not able to provide either explanations or insights.

The code on the online environment always shows true and false values results for each run. We were unsuccessful in implementing a way in which this output might not be printed, so the program runs and prints as an output the English word equivalent for the provided N, and true. In case for N > 1000 the output is only false.