Assignment #2 on Prolog

Date Due: October 25, 2024

Total: 100 marks

Include for all programs both rules/code and execution. Don't use code from the internet. Write your own code.

Please use only the type of constructions that you see the slides/videos/examples. Don't use built-in predicates that will solve the problem for you[1]; that will defeat the purpose of the assignment.
Don't use predicates that emulate procedural programming constructions like for loops(any of them: logical or counter control loops) or conditionals(all kind). Use just the BNF syntax present in the slides – no other extensions. You need to think in Prolog, not in a procedural programming language. By using procedural programming constructions you'll get 10% of the mark or 0.
The content of the slides is enough to complete the assignment. You must send the code(text) together with intructions of how to run the programs (the `Readme` file), and a text capturing the execution (the `Run.txt` file) (no binary files!)[2]. Just use the general format required for all assignemnt submissions, as it is described in the slides.

1. (10 marks) Write the corresponding program for the following predicate:

   ```
   translate(L1,L2).
   ```

   to translate a list words naming digits between 0 and 9 into a list of corresponding digits.

   For example:

   ```
   translate([one,two,nine],[1,2,9]).
   ```

2. (15 marks) Write a solution for a function/predicate that computes the shuffle of two lists, which is defined bellow[3]:

[1]Some versions of Prolog may have that predicates built in
[2]pictures/screenshots are binary files
[3]Warning: don't even try to think to use code from the internet, or any kind of AI

```
|? - shuffle([a,b,c],[1,2,3],X).

X = [a,b,c,1,2,3] ? ;

X = [a,b,1,c,2,3] ? ;

X = [a,1,b,c,2,3] ? ;
....
X = [1,2,3,a,b,c]

yes
```

The shuffle operation is formally defined as:

(a) $\varepsilon \sqcup\!\sqcup \varepsilon = \varepsilon$ [4]

(b) $wa \sqcup\!\sqcup vb = (wa \sqcup\!\sqcup v)b \cup (w \sqcup\!\sqcup vb)a$, where a and b are letters and w and v are lists, possibly empty[5].

3. (15 marks) Write a predicate to interactively guess an odd number from 2000 to 4000 in at most 6 tries. In case the user types an even number, the number of tries is reduced by an additional try. After each try, the number of remaining tries should be displayed. Outputs can be obtained with predicates `write(X)` and `nl`, and an input with the predicate `read(X)` (integer input will be followed by a "."). The secret number should be hardcoded.

   For example:

```
game.
You have 6 tries to guess a number between 2000 and 4000
6.
You have 4 tries to guess a number between 2000 and 4000
3001.
The number is lower, 3 tries remaining.
2501.
You guessed right.
yes.
```

4. (10 marks) Write a predicate to interactively guess an element from two lists (stored internally) in at most k tries, where k is also an input (integer) value. The program should precalculate the maximum and minimum number from both lists and should tell the user if the number is out of bounds. Therefore, you input k followed by at most k tries. The predicate will be true if you guess a number from either lists and false otherwise.

[4] ε is the empty word, the word with no letters. In your case that's the empty list.

[5] Don't use other definitions or interpretations of the English word shuffle; natural language is ambiguous by definition

For example:

```
game.
Number of tries:
4.
You have 4 tries to guess a number from the two lists.
60.
You have 3 tries to guess a number from the two lists.
3001.
The number is out of bounds, 2 tries remaining.
2502.
You guessed right.
```

5. (15 marks) Write a predicate `sublistxy(L,S)` that takes a list `L` on the first position as input, reads a number `X` from the standard input and produces the list `S` on the second position as a sublist of `L` in between position `X` and the end of the list, inclusive (the easier version).

   Counting the positions in `L` starts with 1. In case `X` is less than 1, the selection starts with the first position. In case `X` is greater than the length of the list `L`, the result `S` is the empty list.

   Example:

```
sublist([10,20,30],S).
position:
2.
yes.
S=[20,30]
sublist([10,20,30],S).
position:
4.
yes.
S=[]
sublist([1,2,3],S).
position:
-2.
yes.
S=[1,2,3]
```

6. (20 marks) Write a predicate that computes the sum of all odd numbers contained in a list `L` and are between two other natural numbers. The list is hardcoded internaly, but the two numbers are read from standard input.

7. (maximum 25 marks)

   (a) (10 marks) Write two polynomial functions of your choice, f and g. For each of them write the predicates $pf(X, f(X))$, and $pg(X, g(X))$, that are true for any instantiation of the variable X.

   (b) (20 marks) For two pairs of functions as in 7a write the following. Write a predicate that determines if two numbers are in the following relationship: $a = f(b)$, $b = g(a)$, where, f and g defined as above. You must choose your own functions f and g and you may limit the interval, which can be a discreet one[6], where you search the values a and b [7].

8. (25 marks) Write a predicate to interactively compute the price for a menu. You should have at least 5 questions for menu choices, accept quantities, then compute the total amount and display it. In case the quantity required for a certain item exceeds the amount available, it should be adjusted to the maximum available and the user warned about the adjustment being made.

[6]For example you can search for all numbers in interval $[0, 3]$, and the number have ony one decimal. In this case you will test the values: $0.0.1, 0.2, 0.3, \ldots, 2.9, 3$.

[7]Be careful as prolog has rounding errors