**PROLOG Programming Tasks – 22nd February 2023**

**Implement all**

Exercise 2.1. Write a Prolog predicate analyse\_list/1 that takes a list as its argument and prints out the list’s head and tail on the screen. If the given list is empty, the predicate should put out a message reporting this fact. If the argument term isn’t a list at all, the predicate should just fail. Examples:

?- analyse\_list([dog, cat, horse, cow]).

This is the head of your list: dog

This is the tail of your list: [cat, horse, cow]

Yes

?- analyse\_list([]).

This is an empty list.

Yes

?- analyse\_list(sigmund\_freud).

No

Exercise 2.2. Write a Prolog predicate membership/2 that works like the built-in predicate member/2 (without using member/2).

*Hint:* This exercise, like many others, can and should be solved using a recursive approach and the head/tail-pattern for lists.

Exercise 2.3. Implement a Prolog predicate remove\_duplicates/2 that removes all duplicate elements from a list given in the first argument and returns the result in the second argument position. Example:

?- remove\_duplicates([a, b, a, c, d, d], List).

List = [b, a, c, d]

Yes

Exercise 2.4. Write a Prolog predicate reverse\_list/2 that works like the built-in predicate reverse/2 (without using reverse/2). Example:

?- reverse\_list([tiger, lion, elephant, monkey], List).

List = [monkey, elephant, lion, tiger]

Yes

**Exercise 2.7.** Write a predicate replace/4 to replace all occurrences of a given element (second argument) by another given element (third argument) in a given list (first argument). Example:

?- replace([1, 2, 3, 4, 3, 5, 6, 3], 3, x, List).

List = [1, 2, x, 4, x, 5, 6, x]

Yes

**Exercise 2.8.** Prolog lists without duplicates can be interpreted as sets. Write a program that given such a list computes the corresponding power set. Recall that the power set of a set S is the set of all subsets of S. This includes the empty set as well as the set S itself. Define a predicate power/2 such that, if the first argument is instantiated with a list, the corresponding power set (i.e., a list of lists) is returned in the second position. Example:

?- power([a, b, c], P).

P = [[a, b, c], [a, b], [a, c], [a], [b, c], [b], [c], []]

Yes

*Note:* The order of the sub-lists in your result doesn’t matter

**Exercise 3.3.** Write a Prolog predicate fibonacci/2 to compute the nth Fibonacci number. The Fibonacci sequence is defined as follows: F0 = 0 F1 = 1 Fn = Fn−1 + Fn−2 for n > 2 Examples:

?- fibonacci(1, X).

X = 1

Yes

?- fibonacci(3, X).

X = 2

Yes

?- fibonacci(7, X).

X = 13

Yes

While some authors define the sequence slightly differently (with F0 = 1), your implementation should conform to the definition given above.