AIT Semantic and Declarative Technologies Course

Homework P4

For each problem, write a Prolog predicate that corresponds to the provided head comment.

You are free to make use of the predicates defined in the slides and in earlier exercise solutions. Do not use library predicates that are not discussed in the slides.

You can define helper predicates. Try to provide the most accurate head comment (specification, in other words) for the helper predicates. Remember that a head comment is an English language sentence which describes the logical relationship between the arguments of the predicate.

1. Chopping a list

```
% chop(+N, +List, -LofLists): LofLists is a list whose elements are
% nonempty lists, such that the concatenation of these results in List.
% All elements of LofLists, except for the last, have length N, the
% length of the last should be between 1 and N (inclusive).

| ?- chop(2, [1,a,b,2,c], LL).            LL = [[1,a],[b,2],[c]] ? ; no
| ?- chop(3, [1,a,b,2,c], LL).            LL = [[1,a,b],[2,c]] ? ; no
| ?- chop(3, [1,a,b,2,c,5], LL).          LL = [[1,a,b],[2,c,5]] ? ; no
| ?- chop(3, [1,a], LL).                  LL = [[1,a]] ? ; no
| ?- chop(3, [], LL).                     LL = [] ? ; no
```

   Hint: use `split/4` from Homework P3.

2. Enumerating sublists

```
% list_sub(+Whole, ?Part, ?Before, ?Length, ?After): Part is a
sublist of Whole such that there are Before number of elements in
Whole before Part, After number of elements in Whole after Part
and the length of Part is Length.

| ?- list_sub([a,b], Part, Before, Length, After).
Part = [], After = 2, Before = 0, Length = 0 ? ;
Part = [a], After = 1, Before = 0, Length = 1 ? ;
Part = [a,b], After = 0, Before = 0, Length = 2 ? ;
Part = [], After = 1, Before = 1, Length = 0 ? ;
Part = [b], After = 0, Before = 1, Length = 1 ? ;
Part = [], After = 0, Before = 2, Length = 0 ? ; no
```

   This predicate is available in SICStus `library(lists)` as `sublist/5`. Obviously, you should not use this library predicate.
   Hint: make a declarative, non-recursive solution using the library predicate `append/2` and the BIP `length/2`.
   Warning: The suggested solution results in **very-very** simple code :-).

3. First plateau in a list

   Consider a proper list L of arbitrary ground terms (i.e. terms containing no variables). A sublist P of L is called a plateau, if its length is at least two, all its elements are identical, and it is maximal, i.e. it can not be extended to a longer list of identical elements.

```
% first_plateau(+L, ?A, ?Len, ?Suff): The leftmost plateau in list L
% consists of elements A and its length is Len. Suff is the suffix of L
% after the plateau.

| ?- first_plateau([a,b,2,2,2,a+1,a+1,s(1,2)], A, Len, Suff).
A = 2, Len = 3, Suff = [a+1,a+1,s(1,2)] ? ; no

| ?- first_plateau([a+1,a+1,s(1,2)], A, Len, Suff).
A = a+1, Len = 2, Suff = [s(1,2)] ? ; no
```

   You may re-use the predicate `boring/2`.

4. Enumerating all plateaus in a list

```
% plateau(+L, ?A, ?Len): In list L there is a plateau of Len elements,
% each being A.

| ?- plateau([a,b,2,2,2,a+1,a+1,s(1,2)], A, Len).
A = 2,   Len = 3 ? ;
A = a+1, Len = 2 ? ; no
```

   We suggest to use the predicate `first_plateau/4`.