Week3 - Lab3

Prolog exercises

1. Find the last element of a list.

Example:
```
?- my_last(X,[a,b,c,d]).
X = d
```

2. Reverse a list.
```
% reverse(List, ReversedList)
```

3. Eliminate consecutive duplicates of list elements.
If a list contains repeated elements they should be replaced with a single copy of the element.
The order of the elements should not be changed.

Example:
```
?- compress([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [a,b,c,a,d,e]
```

4. Pack consecutive duplicates of list elements into sublists.
If a list contains repeated elements they should be placed in separate sublists.

Example:
```
?- pack([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [[a,a,a,a],[b],[c,c],[a,a],[d],[e,e,e,e]]
```

5. Duplicate the elements of a list.
Example:
```
?- dupli([a,b,c,c,d],X).
X = [a,a,b,b,c,c,c,c,d,d]
```

6. Split a list into two parts; the length of the first part is given.
Do not use any predefined predicates.

Example:
```
?- split([a,b,c,d,e,f,g,h,i,k],3,L1,L2).
L1 = [a,b,c]
L2 = [d,e,f,g,h,i,k]
```

7. Calculate a the sum of the elements of a given list L.

```
% sum (L, Sum)
% calculates and returns the sum of Sum elements of list L
% are assumed to be elements of the list of numbers
```

8. Check whether a given term represents a binary tree
Write a predicate istree/1 which succeeds if and only if its argument is a Prolog term representing a binary tree.
Example:

```
?- istree(t(a,t(b,nil,nil),nil)).
Yes
?- istree(t(a,t(b,nil,nil))).
No
```

9 . Count the leaves of a binary tree
A leaf is a node with no successors. Write a predicate count_leaves/2 to count them.

```
% count_leaves(T,N) :- the binary tree T has N leaves
```

10. Collect the internal nodes of a binary tree in a list (62)
An internal node of a binary tree has either one or two non-empty successors. Write a predicate internals/2 to collect them in a list.

```
% internals(T,S) :- S is the list of internal nodes of the binary tree T.
```

11. Write in prolog programs for the **set** operations like **union**, **intersection** and **difference.** Sets are represented by lists.

a) % union(S1, S2, U).   % U is union of the sets S1 and S2
b) % intersect(S1, S2, I)   % intersection of the sets S1 in S2
c) % diff(S1, S2, D)   % difference of sets S1 and S2 (is in S1 and not in S2)