Programming in Prolog - List of Exercises #3

1. Write a program that can print all elements of a list

   ```
   ?- print_list([a,b,c]).
   a b c
   ```

2. Write a program that creates a list with first and last elements given.

   ```
   ?- create_list(5,12,S).
   S=[5,6,7,8,9,10,11,12]
   ```

3. Write a program that calculates the mean value of a list of numbers.
4. Write a program that detects if a list contains a number, and returns the number (or numbers) in an argument.
5. Write a program that increments each integer element found in a list. For example:

   ```
   ?- increment_elements([5,6,a,8,b],S).
   S=[6,7,a,9,b]
   ```

6. Write a program that can encapsulate each element of a list as a list. For example:

   ```
   ?- encaps([a,b,1,d,e],S).
   S = [[a],[b],[1],[d],[e]]
   ```

7. Write a program that insert zeros between elements of a list. For example:

   ```
   ?- insert_zeros([1,2,3,4,5],S).
   S = [1,0,2,0,3,0,4,0,5,0]
   ```

8. Write a program that can clone a list:

   ```
   ?- clone([g,6,7],S).
   S = [[g,6,7][g,6,7]]
   ```

9. Write a program that, given a list of elements, modify its Nth element with a given element. For example:

   ```
   ?- modify([m,o,d,i,f,y,e,t],6,i,Y).
   Y = [m,o,d,i,f,y,i,t]
   ```

10. Write a program that generates random integers between I and J, for a square matrix with N rows. For example:

```
% random_matrix(I,J,N,Mat).
?- random_matrix(0,9,3,M).
M = [[2,4,5],[1,0,3],[9,3,2]]
```

11. Consider a representation of sets as lists. Define the following predicates:

    (a) `subset(L,K)`, which holds iff L is a subset of K.

    (b) `disjoint(L,K)`, which holds iff L and K are disjoint (i.e. they have no elements in common).

    (c) `union(L,K,M)`, which holds iff M is the union of L and K.

    (d) `intersection(L,K,M)`, which holds iff M is the intersection of L and K.

    (e) `difference(L,K,M)`, which holds iff M is the difference of L and K.

    Consider two different implementations: (1) the input list can have repeated elements, (2) the input list does not have repeated elements (it is, in fact, a set).

12. Define a predicate `length(L,N)` which holds iff N is the length of the list L.

13. Define a predicate `sumlist(L,N)` which, given a list of integers L, returns the sum N of all the elements of L. (the input list must contain only numbers.)

14. Define a predicate `add_up_list(L,K)` which, given a list of integers L, returns a list of integers in which each element is the sum of all the elements in L up to the same position. For example:

```
?- add_up_list([1,2,3,4],K).
   K = [1,3,6,10];
   no
```

15. Define a predicate `merge(L,K,M)` which, given two ordered lists of integers L and K, returns an ordered list M containing all the elements of L and K.

16. Consider a representation of binary trees as terms, as follows:

```
emptybt           the empty binary tree
consbt(N,T1,T2)   the binary tree with root N
                  and left and right subtrees T1 and T2
```

    (a) Define a predicate `preorder(T,L)` which holds iff L is the list of nodes produced by the preorder traversal of the binary tree T.

    (b) Define a predicate `search_tree(L,T)` which, given a list of integers L, returns a balanced search-tree T containing the elements of L.