# Preparation for the Colloquy

In this section, we will recap the concepts of:

- Arithmetic Operations
- Operations on Lists (complete, difference, incomplete)
- Deep Lists
- Trees (complete, incomplete)
- Graphs and Side Effects (assert, retract)

## 1 Arithmetic Operations

1. Compute the greatest common divisor of two numbers.
   Ex: ? – gcd(15,25,R).
   R = 5.

2. Compute the least common multiplier of two numbers.
   Ex: ? – lcm(15,25,R).
   R = 75.

3. Compute the divisors of a natural number.
   E.g: ? – divisor(15,R1), divisor(2,R2), divisor(1,R3), divisor(0,R4),divisor(6,R5).
   R1 = [1,3,5,15], R2 = [1,2], R3 = [1], R4 = alot, R5 = [1,2,3,6].

4. Convert a number to binary (the powers of 2 grow from right to left).
   E.g: ? – to_binary(5,R1),to_binary(8,R2),to_binary(11,R3).
   R1 = [1,0,1], R2 = [1,0,0,0], R3 = [1,0,1,1].

5. Reverse a natural number.
   E.g: ? – reverse(15,R1), reverse(121235124,R2).
   R1 = 51, R2 = 421542121.

## 2 Operations on Lists

6. Compute the sum of the elements of a list.
   Ex: ? – sum([1,2,3,4,5], R).
   R = 15.

7. Double the odd numbers and square the even.
   E.g: ? – numbers([2,5,3,1,1,5,4,2,6],R).
   R = [4,10,6,2,2,10,16,4,36].

8. Separate the even elements on odd positions from the rest (the position numbering starts at 1).
   E.g: ? – separate([1,2,2,3,4,5,6,6,12,44,8,5,5,10,5],Even,Rest).
   Even = [2,4,6,12,8], Rest = [1,2,3,6,44,5,5,10,5].

9. Replace all occurrences of X with Y.
   E.g: ? – replace_all(1, a, [1,2,3,1,2], R).
   R = [a,2,3,a,2].

10. Replace all the occurrences of x in a difference list (2nd and 3rd argument) with the sequence [Y,X,Y].
    % replace_all(X, S, E, Y, R). unde lista diferență este S-E = [1,2,3,4,2,1,2]
    Ex: ? – replace_all(2,[1,**2**,3,4,**2**,1,**2**,2,3],[2,3],8,R).
    R = [1,**8,2,8**,3,4,**8,2,8**,1,**8,2,8**].

11. Delete the occurrences of X on even positions (the position numbering starts with 1).
    E.g: ? – delete_pos_even([1,2,3,4,2,3,3,2,5],2,R).
    R = [1,3,4,2,3,3,5].

12. Delete each kth element from the list.
    E.g: ? – delete_kth([6,5,4,3,2,1], 3, R).
    R = [6,5,3,2].

13. Delete each kth element from the end of the list.
    E.g: ? – delete_kth_end([1,2,3,4,5,6,7,8,9,10],3,R)
    R = [1,3,4,6,7,9,10].

14. Delete all occurrences of the minimum/maximum element in a list.
    E.g: ? – delete_min([4,5,1,2], R).
    R = [4,5,2].

15. Delete duplicate elements from a list (keep first or last occurrence).
    E.g: ? – delete_duplicates([3,4,5,3,2,4], R).
    R = [3,4,5,2]. sau R = [5,3,2,4].

16. Revese an incomplete list.
    E.g: ? – reverse([1, 2, 3, 4, 5|_], R).
    R = [5, 4, 3, 2, 1|_].

17. Reverse the elements of a list after position K.
    E.g: ? – reverse_k([1,2,3,4,5,6], 2, R).
    R = [1,2,6,5,4,3].

18. Encode a list with RLE.
    E.g: ? – rle_encode([a,a,a,a,b,c,c,a,a,d,e,e,e,e], R).
    R = [[a,4], [b,1] ,[c,2], [a,2], [d,1] , [e,4]].

19. Decode a list encoded with RLE.
    E.g: ? – rle_decode([[a,4], [b,1] ,[c,2], [a,2], [d,1] , [e,4]],R).
    R = [a,a,a,a,b,c,c,a,a,d,e,e,e,e].

20. Rotate a list K positions to the right.
    E.g: ? – rotate_k([1,2,3,4,5,6|_], 2, R).
    R = [5,6,1,2,3,4|_].

21. Sort a list of characters by their ASCII codes.
    E.g: ? – sort_chars([e, t, a, v, f], R).
    R = [a, e, f, t, v].

22. Sort a list of lists by the length of the lists on the second level.
    E.g: ? - sort_len([[a, b, c], [f], [2, 3, 1, 2], [], [4, 4]], R).
    R = [[], [f], [4, 4], [a, b, c], [2, 3, 1, 2]].

23. Delete duplicate elements that are on an odd position in a list (the position numbering starts at 1).
    E.g: ? – remove_dup_on_odd_pos([1,2,3,1,3,3,3,9,10,6,10,8,7,3],R).
    R = [2,1,3,9,6,8,7,3].

# 3 Deep Lists

24. Compute the maximum depth of a deep list.
    E.g: ? – depth([1, [2, [3]], [4]], R1), depth ([], R2).
    R1 = 3, R2 = 1.

25. Flatten a deep list with incomplete lists.
    E.g: ? - flat([[1|_], 2, [3, [4, 5|_]|_]|_], R).
    R = [1,2,3,4,5|_].

26. Flatten only the elements at depth X from a deep list.
    E.g: ? – flatten_only_depth([[1,5,2,4],[1,[4,2],[5,[6,7,8]]],[4,[7]],8,[11]],3,R).
    R = [4,2,5,7].

27. Compute the sum of all element at depth K in a deep list.
    E.g: ? – sum_k([1, [2, [3|_]|_], [4|_]|_], 2, R).
    R = 6.

28. Count the number of lists in a deep list.
    E.g: ? – count_lists([[1,5,2,4],[1,[4,2],[5]],[4,[7]],8,[11]],R).
    R = 8.

29. Replace all occurrences of X with Y in a deep list.
    E.g: ? – replace_all(2, 5, [[1, [2, [3, 2]], [4]], R).
    R = [1, [5, [3, 5]], [4]].

30. Replace each constant depth sequence in a deep list with its length.
    E.g: ? – len_con_depth([[1,2,3],[2],[2,[2,3,1],5],3,1],R).
    R = [[3],[1],[1,[3],1],2].

# 4 Trees

31. Compute the depth of a binary complete/incomplete tree.
    E.g: tree(t(6, t(4, t(2, nil, nil), t(5, nil, nil)), t(9, t(7, nil, nil), nil))).
    ? – tree(T), depth(T, R).
    R = 3.

32. Collect all nodes of binary complete/incomplete tree in inorder using complete lists.
    E.g: tree(t(6, t(4, t(2, nil, nil), t(5, nil, nil)), t(9, t(7, nil, nil), nil))).
    ? – tree(T), inordine(T, R).
    R = [2,4,5,6,7,9].

33. Collect all leaves of a binary tree.
    E.g: tree(t(6, t(4, t(2, nil, nil), t(5, nil, nil)), t(9, t(7, nil, nil), nil))).
    ? – tree(T), collect_k(T, R).
    R = [2,5,7].

34. Write a predicate which checks whether the tree is a binary search tree.
    E.g: tree(t(3, t(2, t(1, nil, nil), t(4, nil, nil)), t(5, nil, nil))).
    ? – tree(T), is_bst(T).
    false.

35. Binary incomplete tree. Collect odd nodes with 1 child in an incomplete list.
    E.g: tree(t(26,t(14,t(2,_,_),t(15,_,_)),t(50,t(35,t(29,_,_),_),t(51,_,t(58,_,_))))).
    ? – tree(X), collect_odd_from_1child(X,R).
    R = [35, 51|_].

36. Ternary incomplete tree. Collect the keys between X and Y (closed interval) in a difference list.
    E.g: tree(t(2,t(8,_,_,_),t(3,_,_,t(4,_,_,_)),t(5,t(7,_,_,_),t(6,_,_,_),t(1,_,_,t(9,_,_,_))))).
    ? – tree(T), collect_between(T,2,7,R,[1,18]).
    R = [2,3,4,5,6,7,1,18].

37. Binary Tree. Collect even keys from leaves in a difference list.
    E.g: tree(t(5,t(10,t(7,nil,nil),t(10,t(4,nil,nil),t(3,nil,t(2,nil,nil)))),t(16,nil,nil))).
    ? – tree(T), collect_even_from_leaf(T,R,[1]).
    R = [4,2,16,1].

38. Replace the min element from a ternary incomplete tree with the root.
    E.g: tree(t(2,t(8,_,_,_),t(3,_,_,t(1,_,_,_)),t(5,t(7,_,_,_),t(6,_,_,_),t(1,_,_,t(9,_,_,_))))).
    ? – tree(T), replace_min(T,R).
    R = t(2,t(8,_,_,_),t(3,_,_,t(2,_,_,_)),t(5,t(7,_,_,_),t(6,_,_,_),t(2,_,_,t(9,_,_,_))))5

39. Collect all nodes from depth K in a binary tree.
    Ex: tree(t(6, t(4, t(2, nil, nil), t(5, nil, nil)), t(9, t(7, nil, nil), nil))).

? – tree(T), collect_k(T, 2, R).
R = [4, 9].

40. Collect all the nodes at odd depth from a binary incomplete tree (the root has depth 0).
E.g: tree(t(26,t(14,t(2,_,_),t(15,_,_)),t(50,t(35,t(29,_,_),_),t(51,_,t(58,_,_))))).
? – tree(X), collect_all_odd_depth(X,R).
R = [14,50,29,58].

41. Collect the subtrees having the median value from a ternary incomplete tree.
*Note. The median is the "middle" of the sorted list of keys.*
E.g: tree(t(2,t(8,_,_,_),t(3,_,_,t(1,_,_,_)),t(5,t(7,_,_,_),t(5,_,_,_),t(1,_,_,t(9,_,_,_))))).
? – tree(T), median(T,R).
R = [
t(5,t(7,_,_,_),t(5,_,_,_),t(1,_,_,t(9,_,_,_)))),
t(5,_,_,_)
].

42. Replace each node with its height in a binary incomplete tree (a leaf has height 0).
E.g: tree(t(2,t(4,t(5,_,_),t(7,_,_)),t(3,t(0,t(4,_,_),_),t(8,_,t(5,_,_))))).
? – tree(T), height_each(T,R).
R = tree(t(3,t(1,t(0,_,_),t(0,_,_)),t(2,t(1,t(0,_,_),_),t(1,_,t(0,_,_))))).

43. Write a predicate which replaces the entire subtree of a node (whose key is given as argument) with a single node having as key the sum of the keys in the subtree of that node (if there is no such node in the tree, leave the structure unchanged).
E.g: tree(t(14,t(6,t(4,nil,nil),t(12,t(10,nil,nil),nil)),t(17,t(16,nil,nil),t(20,nil,nil)))).
? – tree(T), sum_subtree(T,6,R).
R = t(14,t(32,nil,nil),t(17,t(16,nil,nil),t(20,nil,nil)))).

# 5 Graphs

44. Collect all nodes of a graph.
    Ex: node(1). node(2). node(3).
    ? – collect(R).
    R = [1,2,3].

45. Compute the indegree and the outdegree for each node in a graph using the dynamic predicate info(Node, OutDegree, InDegree).
    E.g: edge(1,2). edge(2,1). edge(1,4). edge(1,3). edge(3,2).
    => info(1,3,1). info(2,1,2). info(3,1,1). info(4,0,1).