1. **Write a PROLOG program to implement a family-tree.**

2. **Using PROLOG, write a series of facts and rules that asserts the facts that Bob and Mary speak Russian and John and Mary speak English. It also defines the relation "understands" between two persons, which is true exactly when they both speak the same language. Your program should answer the following queries:**
    a. **?- speaks(X, Russian).**
    b. **?- understands(John, Bob).**
    c. **?- understands(X, Bob).**
    d. **?- understands(P1, P2).**

3. **Write a PROLOG program to calculate the sum of two numbers.**

4. **Write a PROLOG program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.**

5. **Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.**

6. **Write a program in PROLOG to implement generate_fib(N,T) where T represents the Nth term of the fibonacci series.**

7. **Write a PROLOG program to implement GCD of two numbers.**

8. **Consider a cyclic directed graph [edge (p, q), edge (q, r), edge (q, r), edge (q, s), edge (s,t)] where edge (A,B) is a predicate indicating directed edge in a graph from a node A to a node B. Write a PROLOG program to check whether there is a route from one node to another node.**

9. **Write a PROLOG program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.**

10. **Write a PROLOG program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.**

11. **Write a program in PROLOG to implement towerofhanoi (N) where N represents the number of discs.**

12. **Write a PROLOG program to implement memb(X, L): to check whether X is a member of L or not.**

13. **Write a PROLOG program to implement last_el (L, X) where L is a list and X represents the last element of list L.**

14. **Write a menu-driven PROLOG program to implement the following:**
    a. **conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**
    b. **reverse (L, R) where List L is original and List R is a reversed list.**
    c. **palindrome (L) which checks whether a list L is a palindrome or not.**

15. **Write a PROLOG program to implement nth_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.**

16. **Write a PROLOG program to implement sumlist(L, S) so that S is the sum of a given list L.**

17. **Write a PROLOG program to implement maxlist(L, M) so that M is the maximum number in the list L.**

18. **Write a PROLOG program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.**

19. **Write a PROLOG program to implement delete_first (X, L ,R) where X denotes the element whose first occurrence has to be deleted from list L to obtain list R.**

20. **Write a PROLOG program to implement insert_nth(I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.**

21. **Write a PROLOG program to implement delete_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.**

22. **Write a program in PROLOG to implement remove_dup (L, R) where L denotes the list with some duplicates and the list R denotes the list with duplicates removed.**

23. **Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.**

24. **Write a program in PROLOG to implement permute (L, P) where P represents all possible permutations of the elements of List L.**

25. **Write a program in PROLOG to implement is_subsequence(S,L) so that it returns true if S is a subsequence of list L.**

26. **Write a program in PROLOG to implement is_sublist(S,L) so that it returns true if S is a sublist of list L.**

27. **Write a program in PROLOG to implement delete_all (X, L, R) where X denotes the element who's all occurrences has to be deleted from list L to obtain list R.**

28. **Write a program in PROLOG to implement inorder(T), preorder(T) and postorder(T), where T represents a tree, to print inorder, preorder and postorder traversals of the tree.**

29. **Write a program in PROLOG to implement Depth-First-Search of a tree.**

30. **Write a PROLOG program that will take grammar rules in the following format:**

    NT (NT | T)*

    Where NT is any nonterminal, T is any terminal and Kleene star (*) signifies any number of repetitions, and generate the corresponding top-down parser, that is:

    sentence → noun-phrase, verb-phrase

    determiner → [the]

    will generate the following:

    sentence (I, O) :- noun-phrase(I,R), verb-phrase (R,O).

    determiner ([the|X], X) :- !.