|  |
| --- |
| /\***Practical 11** |
|  | Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.\*/ |
|  | palindrome(L):- |
|  | reverse(L,L). |
|  |  |
|  | /\***Practical 12** |
|  | Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.\*/ |
|  | sumlist([],0). |
|  | sumlist([X|L1],S):- |
|  | sumlist(L1,S1), |
|  | S is S1+X. |
|  |
| /\***Practical 13** |
|  | 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.\*/ |
|  | evenlength([]). |
|  | evenlength([\_|T]):- |
|  | oddlength(T). |
|  | oddlength([\_|T]):- |
|  | evenlength(T). |
|  |  |
|  | /\***Practical 14** |
|  | 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.\*/ |
|  | nth\_element(1, [X|\_], X). |
|  | nth\_element(K,[\_|L],X):- |
|  | nth\_element(K1,L,X), |
|  | K is K1+1. |
|  |  |
|  | /\***Practical 15** |
|  | Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.\*/ |
|  | maxlist([X],X). |
|  | maxlist([H|T],M):- |
|  | maxlist(T,M1), |
|  | H<M1 -> M is M1; |
|  | M is H. |
|  |  |
|  | /\***Practical 16** |
|  | 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.\*/ |
|  | insert\_nth(I, 1, L, [I|L]). |
|  | insert\_nth(I, N, [H|T], [H|R]):- |
|  | N1 is N-1, |
|  | insert\_nth(I, N1, T, R). |
|  |  |
|  | /\***Practical 17** |
|  | 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.\*/ |
|  | delete\_nth(1, [\_|T], T). |
|  | delete\_nth(N, [H|T], [H|R]):- |
|  | N1 is N-1, |
|  | delete\_nth(N1, T, R). |
|  |  |
|  | /\***Practical 18** |
|  | 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.\*/ |
|  | merge([],L2,L2). |
|  | merge(L1,[],L1). |
|  | merge([H1|T1],[H2|T2],[H1|T3]):- |
|  | H1=<H2, |
|  | merge(T1, [H2|T2], T3). |
|  | merge([H1|T1],[H2|T2],[H2|T3]):- |
|  | merge([H1|T1], T2, T3). |
|  |  |
|  |  |