CS4271 Artificial Intelligence Lab

Assignment 1

Name: Abhiroop Mukherjee

Enrolment No.:510519109

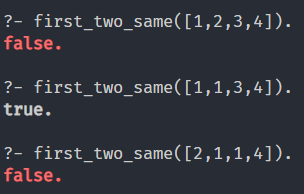
GSuite ID: [510519109.abhirup@students.iiests.ac](mailto:510519109.abhirup@students.iiests.ac).

## Entire Code

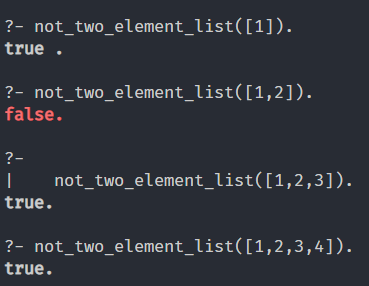
| % 1. To determine whether the first two elements of a list  % are same first\_two\_same([X|[X|\_]]).  % 2. To determine whether a list is not a two-element list. not\_two\_element\_list([]). not\_two\_element\_list([\_]). not\_two\_element\_list([\_| [\_|[\_ | \_]]]).  % 3. To determine whether two lists are of same length. equal\_length([],[]). equal\_length([\_ | L1], [\_ | L2]):-  equal\_length(L1, L2).  % 4. To determine length of a list using your own number system. % length 0 -> 0 % length 1 -> s(0) % length 2 -> s(s(0)) my\_list\_length([], 0). my\_list\_length([\_| Rest], s(L)):-  my\_list\_length(Rest, L).  % 5. To determine whether two lists are of same length using  % the length predicate developed in 4 (previous problem). equal\_length1(L1, L2):-  my\_list\_length(L1, X),  my\_list\_length(L2, X).  % 6. To find the last element of a list.  last\_element([X], X). last\_element([\_ | [B |L1]], X):-  last\_element([B | L1], X).  % 7. To find whether an element is a member of a list. is\_member(X, [X]). is\_member(X, [\_ | Rest]):-  is\_member(X, Rest).  % 8. To find whether two elements are next to each other in  % a list. two\_elem\_next(A, B, [A | [B | \_]]):- !. two\_elem\_next(A, B, [\_ | Rest]):-  two\_elem\_next(A, B, Rest).  % 9. To append two lists in a third list. % append\_two\_list(L1, L2, L3). -> appends L1 and L2 and puts it in L3 append\_two\_list([], L2, L2).  append\_two\_list([X|L1], L2, [X|L3]):-  append\_two\_list(L1, L2, L3).  % 10. To find the last element of a list using append  % predicate developed in 9 last\_element1(L1, X):-  append\_two\_list(\_, [X], L1).  % 11. To find whether an element is a member of a list using % append predicate developed in 9 % is\_member(X, L1). -> true if X is in L1 is\_member1(X, L1):-  append\_two\_list(\_, [X|\_], L1).  % 12. To find whether two elements are next to each other in a list using append predicate % developed in 9. % next\_each\_other(X, Y, L) next\_each\_other(X, Y, L):-  append\_two\_list(\_, [X|[Y|\_]], L).  % 13. To reverse a list in another list. % reverse\_list(L, R), R is the resultant reverse\_list([], []). reverse\_list([X|L], L1):-  reverse\_list(L, L2),  append\_two\_list(L2, [X], L1).  % 14. To determine whether a list is a palindrome. % is\_palindrome(L) -> give true if palindrome is\_palindrome(L):-  reverse\_list(L, L).  % 15. To find the last but one element of a list. % last\_but\_one(L, X) -> X is the last but one element of list L last\_but\_one(L, X):-  append\_two\_list(\_, [X|[\_]], L).  % 16. To find the sum of all elements of a list. % sum\_list(L, S) -> S is the sum of all elements of list L sum\_list([], 0). sum\_list([H|T], Sum) :-  sum\_list(T, Rest),  Sum is H + Rest.  % 17. To find the maximum number from a list.  % max\_two\_num(X, Y, Z) -> Z is the max number between X and Y max\_two\_num(X, Y, X):-  X >= Y. max\_two\_num(X, Y, Y):-  X =< Y.  % max\_list(L, X) -> X is the maximum number of list L max\_list([X], X). max\_list([X | Rest], Max):-  max\_list(Rest, MaxRest),  max\_two\_num(X, MaxRest, Max).  % 18. To find gcd of two integers. % gcd\_two\_num(X, Y, Z) -> Z is the gcd of X and Y gcd\_two\_num(X,Y,G) :- X=Y, G=X. gcd\_two\_num(X,Y,G) :- X<Y, Y1 is Y-X, gcd\_two\_num(X,Y1,G). gcd\_two\_num(X,Y,G) :- X>Y ,gcd\_two\_num(Y,X,G).  % 19. To determine whether a given integer number is prime. % divisible(X, Y), X is divisible by Y divisible(X,Y) :- 0 is X mod Y, !. divisible(X,Y) :- X > Y+1, divisible(X, Y+1). % is\_prime(X) -> true if X is prime is\_prime(2) :- true,!. is\_prime(X) :- X < 2,!,false. is\_prime(X) :- not(divisible(X, 2)).  % 20. To determine whether two positive integer numbers are co-prime. % coprime(X, Y) -> true if X and Y are coprime coprime(X, Y):-  gcd\_two\_num(X, Y, 1).  % 21. To determine the prime factors of a given positive integer. prime\_factors(N, L) :-  findall(D, prime\_factor(N, D), L).  prime\_factor(N, D) :-  find\_prime\_factor(N, 2, D).  find\_prime\_factor(N, D, D) :-  0 is N mod D. find\_prime\_factor(N, D, R) :-  D < N,  (0 is N mod D  -> (N1 is N/D, find\_prime\_factor(N1, D, R))  ; (D1 is D + 1, find\_prime\_factor(N, D1, R))  ).  % 22. Goldbach's conjecture. % goldbach(X, L)-> L is a list of two prime numbers whose sum is X next\_prime(P,P1) :- P1 is P + 2, is\_prime(P1), !. next\_prime(P,P1) :- P2 is P + 2, next\_prime(P2,P1).  goldbach(4,[2,2]). goldbach(N,L) :-  N mod 2 =:= 0,  N > 4,  goldbach(N,L,3). goldbach(N,[P,Q],P) :-  Q is N - P,  is\_prime(Q), P < Q. goldbach(N,L,P) :-  P < N,  next\_prime(P,P1),  goldbach(N,L,P1).  % 23. To count numbers greater than 100.0 in a list. % num\_greater\_than\_100(L, X) -> X are the number of terms % in list L that are greater than 100  num\_greater\_than\_100([], 0).  num\_greater\_than\_100([Head| Rest], X):-  Head > 100, !,   num\_greater\_than\_100(Rest, Y),  X is Y + 1.  num\_greater\_than\_100([\_| Rest], X):-  num\_greater\_than\_100(Rest, X).  % 24. To split a list of numbers in two lists such that  % one contains negative numbers and other contains positive numbers. % split(L, X, Y) -> split L into X have +ve and Y havign -ve split([],[],[]).  split([Head|Tail], [Head|L1], L2):-  Head >= 0, !,  split(Tail, L1, L2).  split([Head|Tail], L1, [Head|L2]):-  split(Tail, L1, L2). |
| --- |

## Output

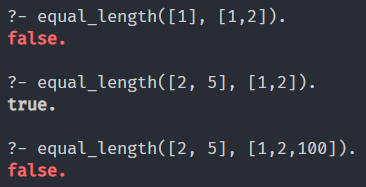
1. To determine whether the first two elements of a list are same.



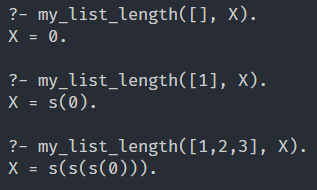
1. To determine whether a list is not a two-element list.



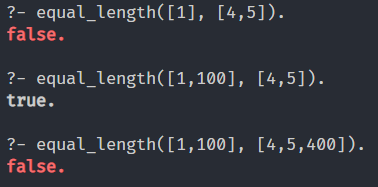
1. To determine whether two lists are of same length.



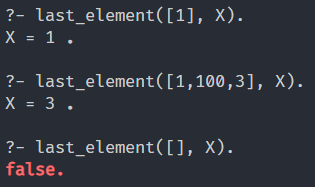
1. To determine length of a list using your own number system.



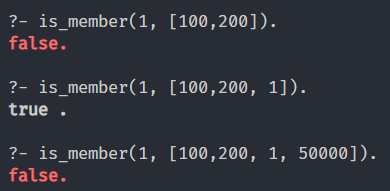
1. To determine whether two lists are of same length using the length predicate developed in 4 (previous problem).



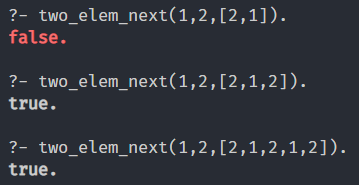
1. To find the last element of a list.



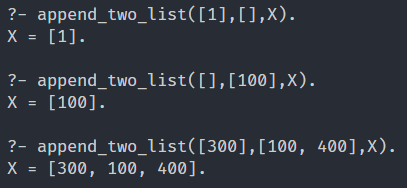
1. To find whether an element is a member of a list.



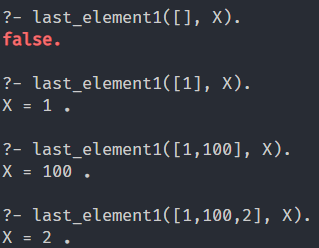
1. To find whether two elements are next to each other in a list.



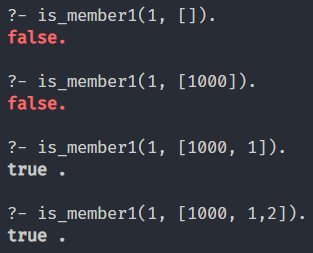
1. To append two lists in a third list.



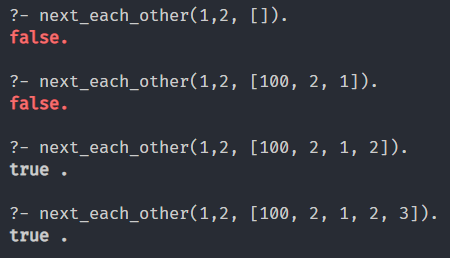
1. To find the last element of a list using append predicate developed in 9



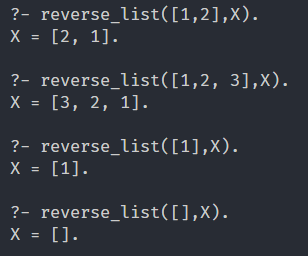
1. To find whether an element is a member of a list using append predicate developed in 9



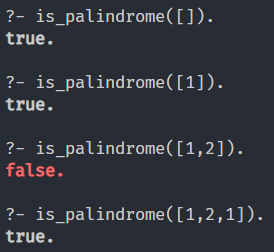
1. To find whether two elements are next to each other in a list using append predicate developed in 9.



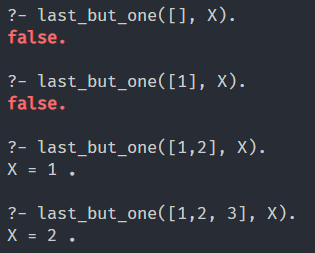
1. To reverse a list in another list.



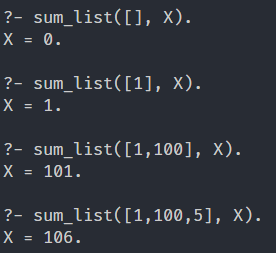
1. To determine whether a list is a palindrome.



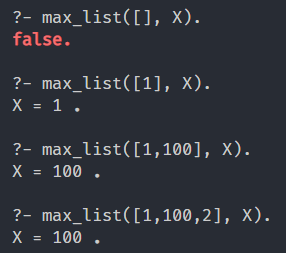
1. To find the last but one element of a list.



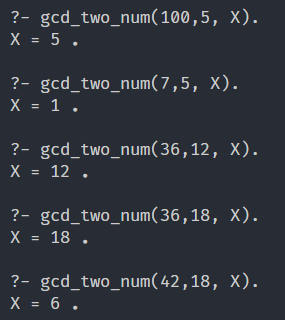
1. To find the sum of all elements of a list.



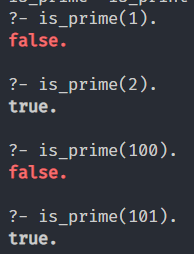
1. To find the maximum number from a list.



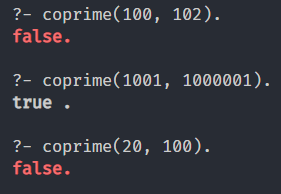
1. To find gcd of two integers.



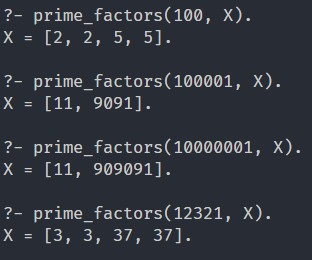
1. To determine whether a given integer number is prime.



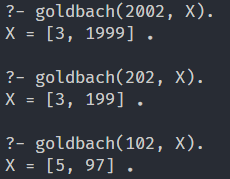
1. To determine whether two positive integer numbers are co-prime.



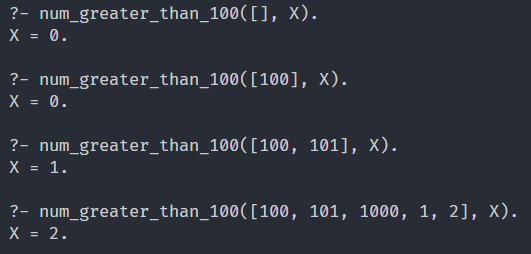
1. To determine the prime factors of a given positive integer.



1. Goldbach's conjecture.



1. To count numbers greater than 100.0 in a list.



1. To split a list of numbers in two lists such that one contains negative numbers and other contains positive numbers.

