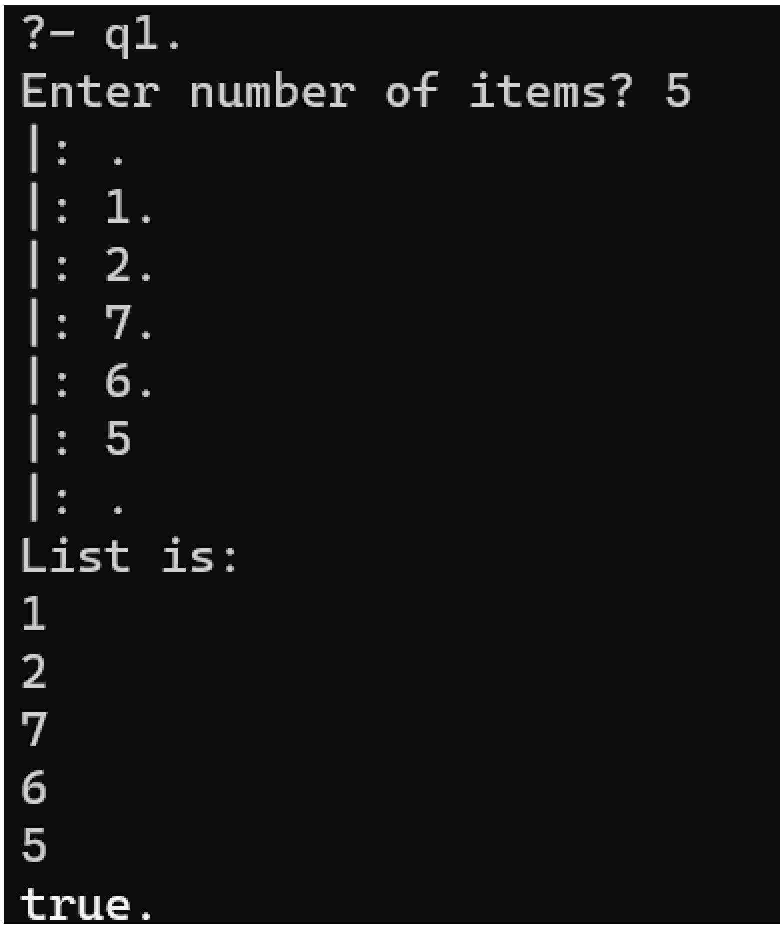
PPL LAB 4

SAHIL BONDRE: U18CO021

1. To input the list from the user and print it (Hint: Use read/2 to input the list).

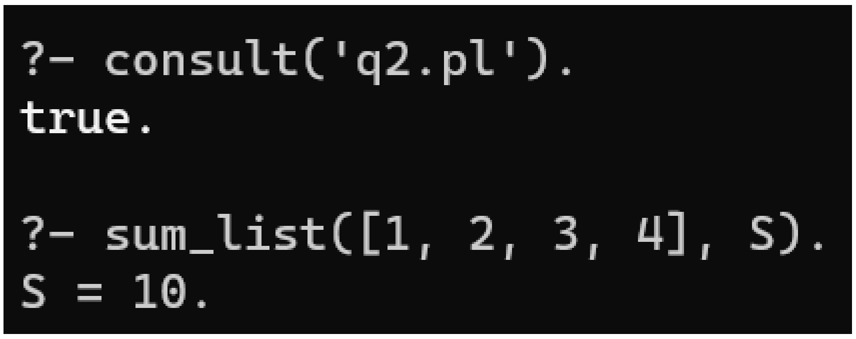
```
print_list([]).

print_list([H|T]):-

  write(H),nl,

  print_list(T).


q1:-

  write('Enter number of items: '),

  read(N),

  length(L, N),

  maplist(read, L),

  write('List is: '), nl,

  print_list(L).
```



```
?- q1.
Enter number of items? 5
|: .
|: 1.
|: 2.
|: 7.
|: 6.
|: 5
|: .
List is:
1
2
7
6
5
true.
```

2. Find the sum of all elements in the list.

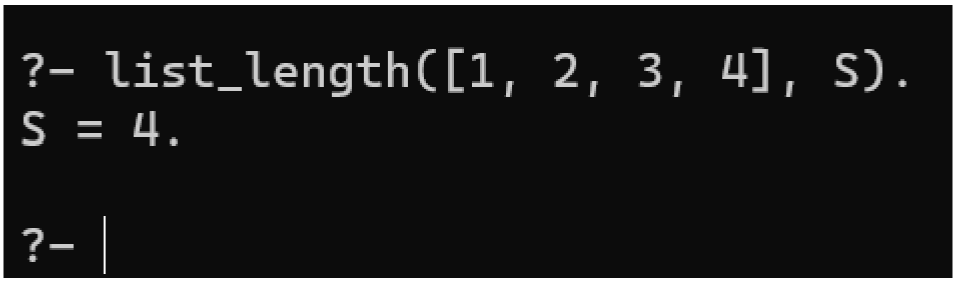
```
sum_list([], 0).

sum_list([H|T], Sum) :-

  sum_list(T, X),

  Sum is H + X.
```



```
?- consult('q2.pl').
true.

?- sum_list([1, 2, 3, 4], S).
S = 10.
```

3. Find the size of a list.

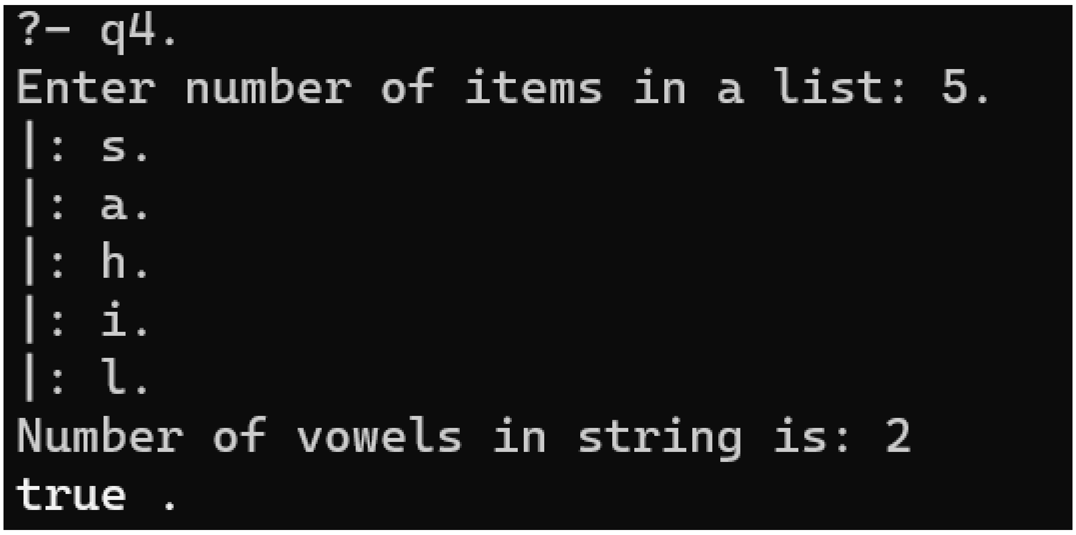
```
list_length([], 0).

list_length([_|T] , L):- list_length(T, N), L is N + 1.
```



```
?- list_length([1, 2, 3, 4], S).
S = 4.

?- 
```

4. Count no. of vowels in a list. (Hint: Input list of characters from a user and count no of vowels in it)

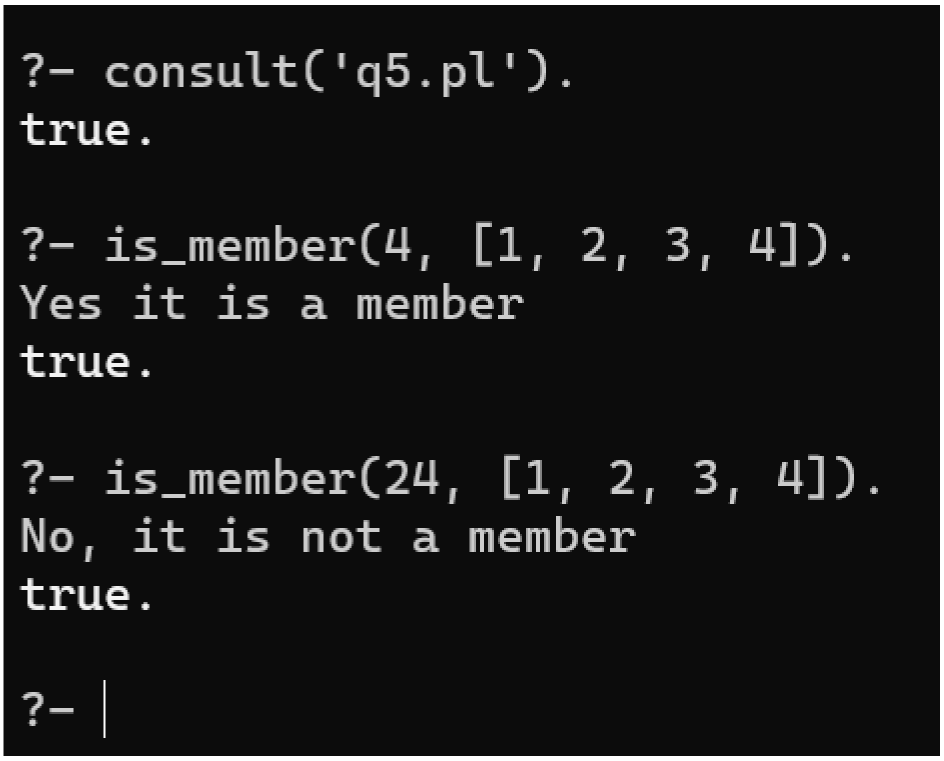
```
vowel(X):- member(X,[a, e, i, o, u]).


vowel_count([], 0).

vowel_count([X|T], N):-

  vowel(X),

  vowel_count(T,N1),

  N is N1+1.


vowel_count([_|T], N):- vowel_count(T, N).


q4 :-

  write('Enter number of items in a list: '),

  read(N),

  length(L, N),

  maplist(read, L),
  vowel_count(L, T),

  write('Number of vowels in string is: '), write(T).
```



```
?- q4.
Enter number of items in a list: 5.
|: s.
|: a.
|: h.
|: i.
|: l.
Number of vowels in string is: 2
true .
```

5. Search whether an element exists in a list.

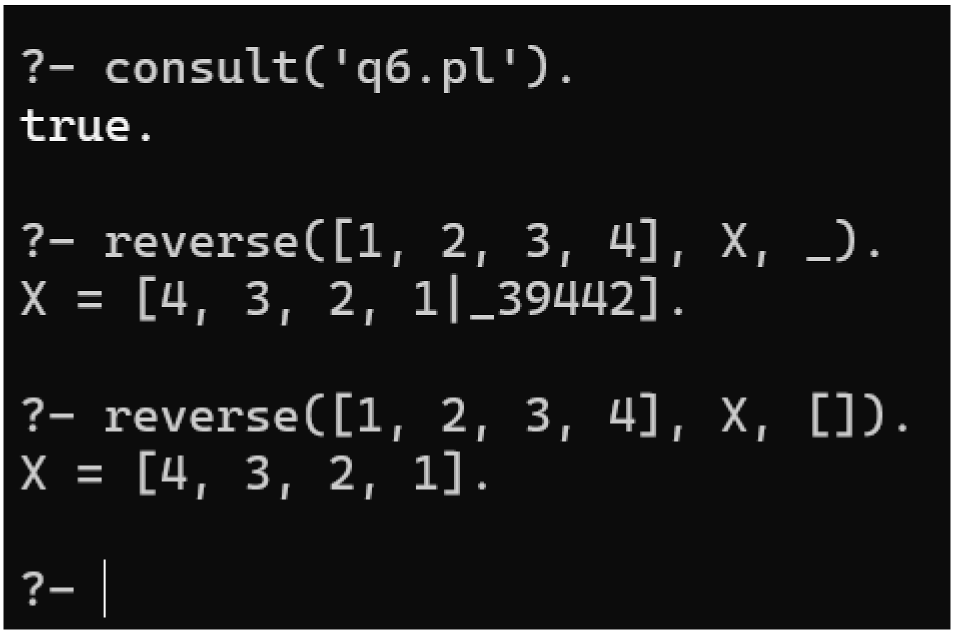
```
is_member(X, [H|T]) :-

(

  member(X, [H|T])->

    write('Yes it is a member'), nl;

    write('No, it is not a member'), nl

).
```



```
?- consult('q5.pl').
true.

?- is_member(4, [1, 2, 3, 4]).
Yes it is a member
true.

?- is_member(24, [1, 2, 3, 4]).
No, it is not a member
true.

?- 
```

6. Reverse a given list.

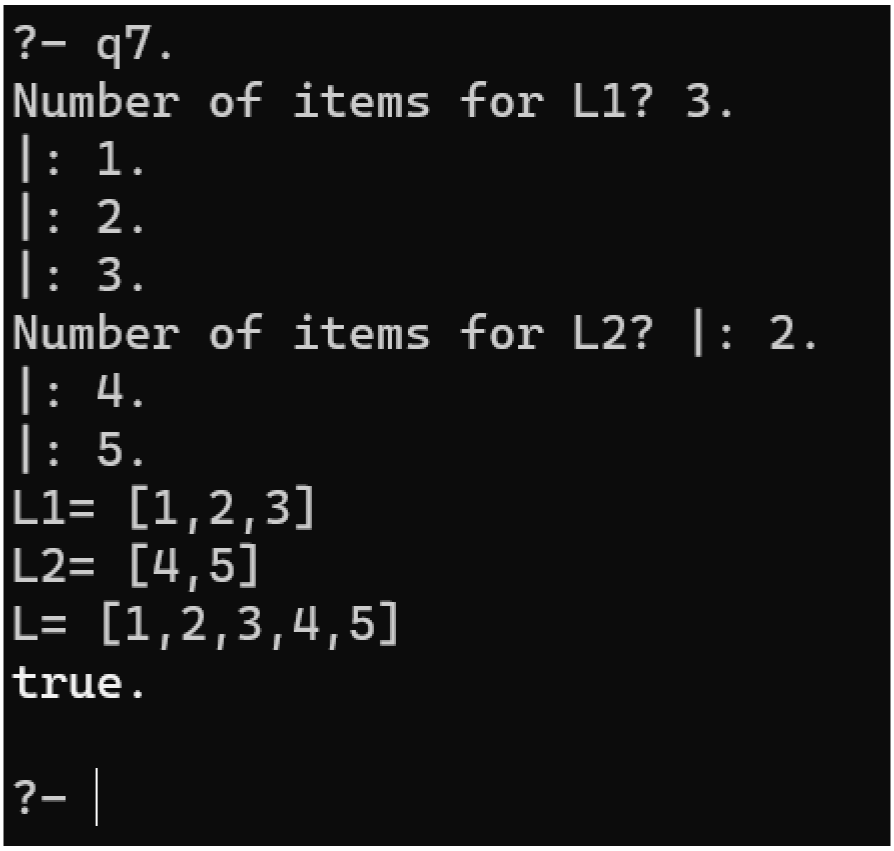
```
reverse([], Z, Z).

reverse([H|T], Z, Temp) :- reverse(T, Z, [H|Temp]).
```



```
?- consult('q6.pl').
true.

?- reverse([1, 2, 3, 4], X, _).
X = [4, 3, 2, 1|_39442].

?- reverse([1, 2, 3, 4], X, []).
X = [4, 3, 2, 1].

?- 
```

7. Concatenate two lists. (Hint: Take two lists namely, L1 and L2 from a user and concatenate it in a list L)

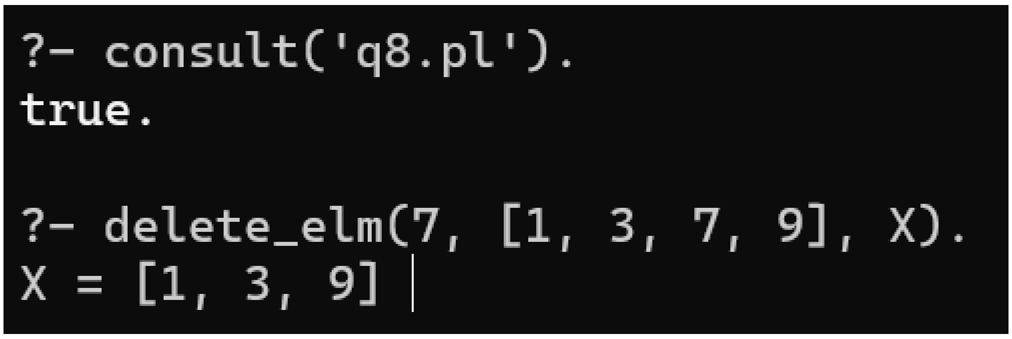
```
concatenate(L1, L2, L):-
  append(L1, L2, L).


q7 :-
  write('Number of items for L1? '),
  read(N),
  length(L1, N),
  maplist(read, L1),
  write('Number of items for L2? '),
  read(N1),
  length(L2, N1),
  maplist(read, L2),
  concatenate(L1, L2, L),
  write('L1= '), write(L1), nl,
  write('L2= '), write(L2), nl,
  write('L= '), write(L).
```



```
?- q7.
Number of items for L1? 3.
|: 1.
|: 2.
|: 3.
Number of items for L2? |: 2.
|: 4.
|: 5.
L1= [1,2,3]
L2= [4,5]
L= [1,2,3,4,5]
true.

?-
```

8. Delete an element from the list.

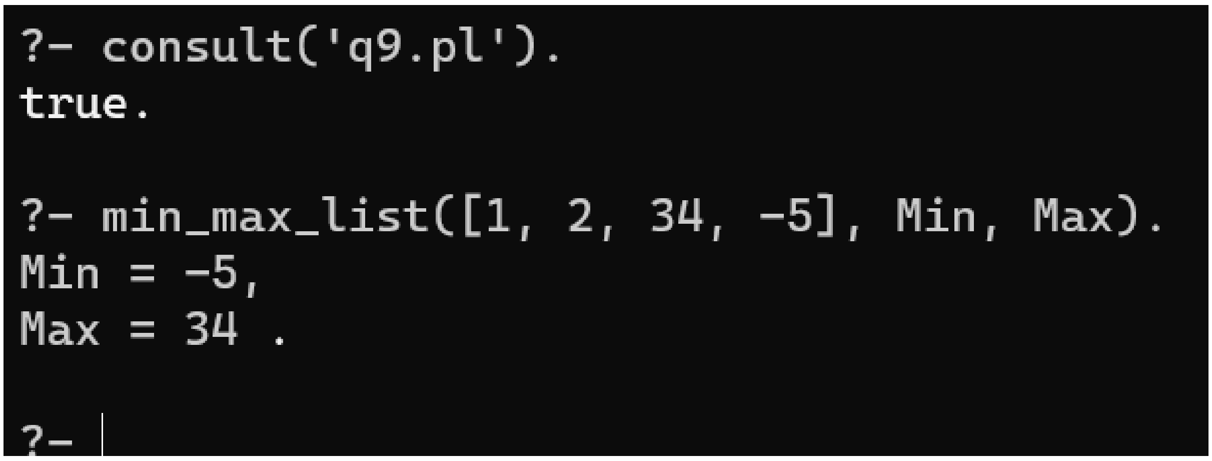
```
delete_elm(Element,[Element|Tail],Tail).


delete_elm(Element,[Head|Tail],[Head|Tail1]) :-

  delete_elm(Element,Tail,Tail1).
```



```
?- consult('q8.pl').
true.

?- delete_elm(7, [1, 3, 7, 9], X).
X = [1, 3, 9]
```

9. Find Max and min elements from the list.

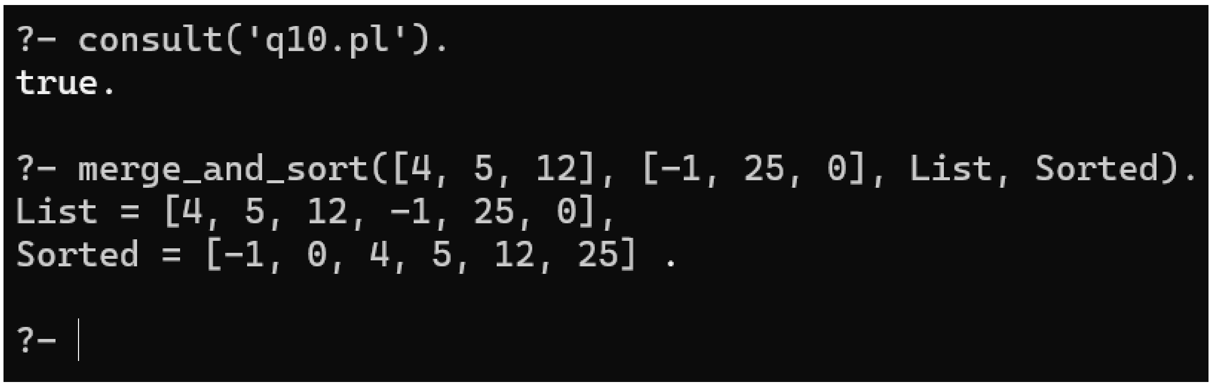
```
min_max_list([A], A, A).


min_max_list([H|R], N, X):-

  min_max_list(R, RN, RX),

  N is min(H, RN),

  X is max(H, RX).
```



```
?- consult('q9.pl').
true.

?- min_max_list([1, 2, 34, -5], Min, Max).
Min = -5,
Max = 34 .

?-
```

10. Merge and sort two given lists in the third list.

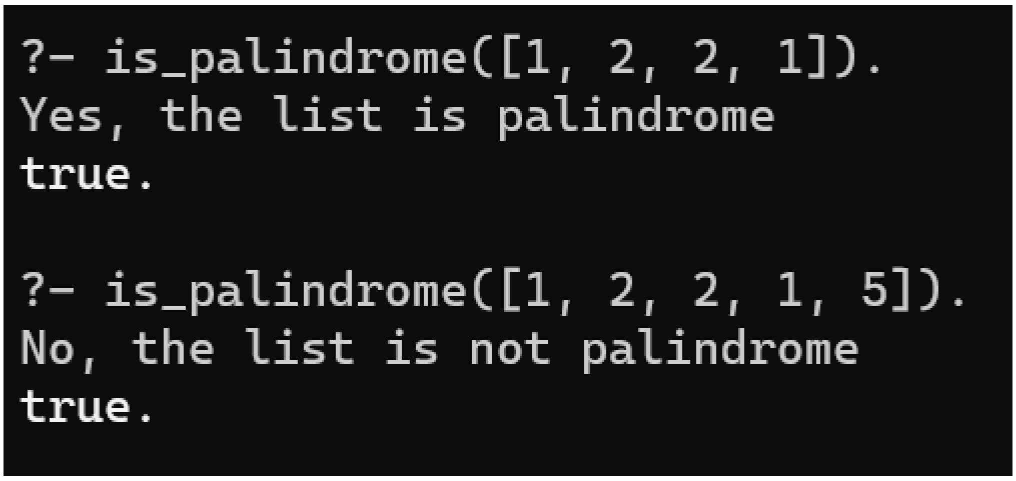
```
sort_list(List, Sorted) :- sort_util(List,[], Sorted).


sort_util([], Acc, Acc).


sort_util([H|T], Acc, Sorted) :- swap_elements(H, T, NT, Max), sort_util(NT,
[Max|Acc], Sorted).


swap_elements(X, [], [], X).


swap_elements(X, [Y|T], [Y|NT], Max) :- X > Y, swap_elements(X, T, NT, Max).
swap_elements(X, [Y|T], [X|NT], Max) :- X =< Y, swap_elements(Y, T, NT, Max).


merge_and_sort(L1, L2, List, Sorted) :-

  append(L1, L2, List),

  sort_list(List, Sorted).
```



```
?- consult('q10.pl').
true.

?- merge_and_sort([4, 5, 12], [-1, 25, 0], List, Sorted).
List = [4, 5, 12, -1, 25, 0],
Sorted = [-1, 0, 4, 5, 12, 25] .

?-
```

11. Check if a given list is a palindrome.

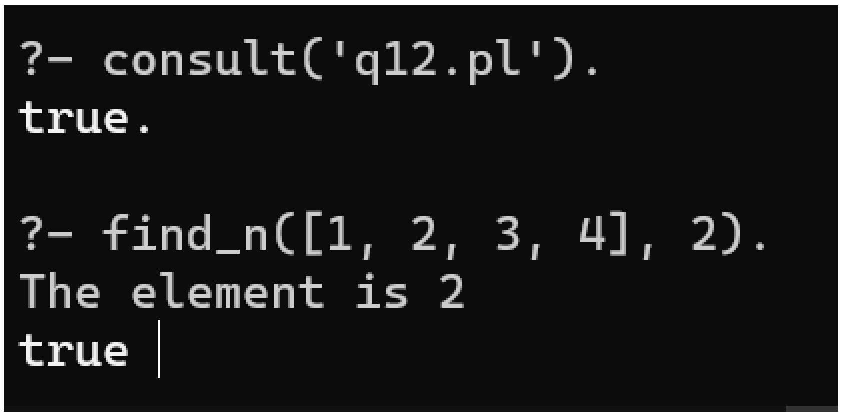
```
palindrome([]).

palindrome([_]).

palindrome(Pal) :-

  append([H|T], [H], Pal),

  palindrome(T).


is_palindrome(L) :-

(

  palindrome(L) -> write('Yes, the list is palindrome'), nl;

    write('No, the list is not palindrome'), nl

).
```



```
?- is_palindrome([1, 2, 2, 1]).
Yes, the list is palindrome
true.

?- is_palindrome([1, 2, 2, 1, 5]).
No, the list is not palindrome
true.
```

12. Find an nth element of the list.

```
find_n([], _):-write('There is no such element in the list'), nl.


find_n([Element|_], 1) :- write('The element is '), write(Element), nl.


find_n([_|List], N) :-

  N1 is N-1,

  find_n(List, N1).
```

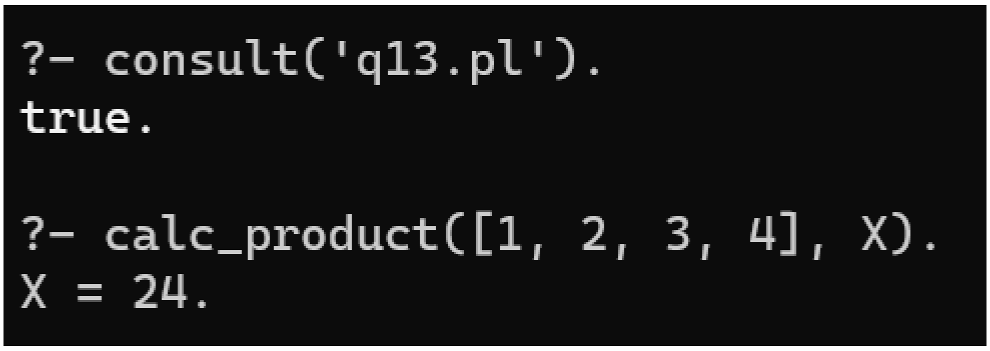


```
?- consult('q12.pl').
true.

?- find_n([1, 2, 3, 4], 2).
The element is 2
true
```

13. Find the product of all elements in the list.

```
product([], 1).

product([H|T], Product):-
```

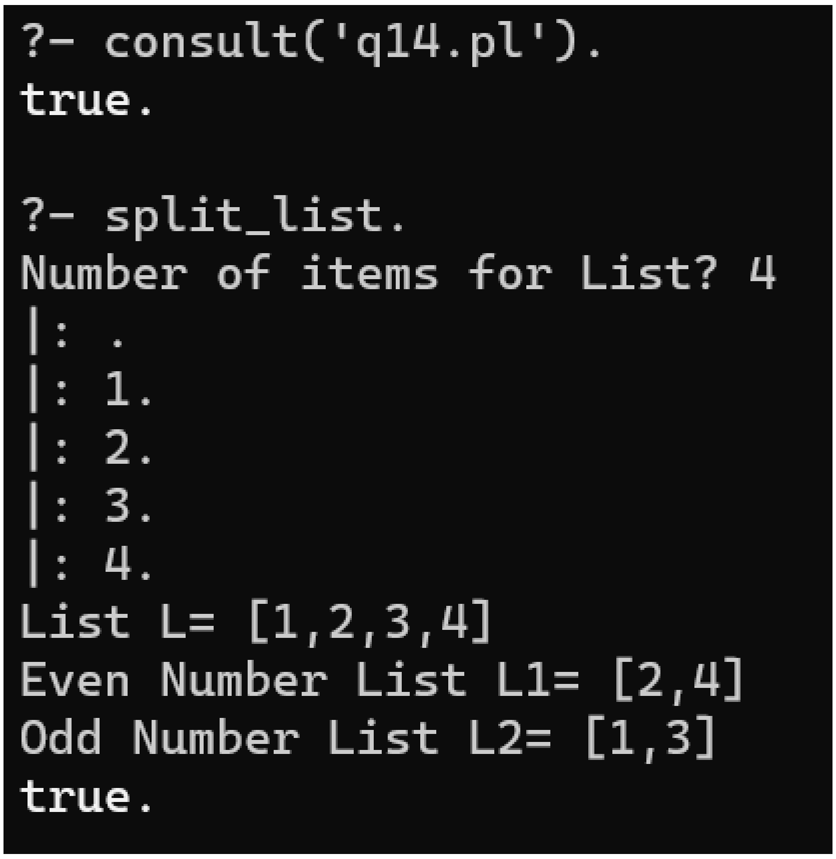
```
  product(T, Rest),

  Product is H * Rest.

  list_length([], 0 ).


list_length([_|Xs] , L ) :- list_length(Xs,N), L is N+1 .


calc_product(L, Product) :-

  list_length(L, Len),

  (

    Len == 0-> Product is 0;

      product(L, Product)

  ).

```



```
?- consult('q13.pl').
true.

?- calc_product([1, 2, 3, 4], X).
X = 24.
```

14. Split the list into two parts. Take list L from the user. The list L1 contains all even elements of the list L and the list L2 contains all odd elements of list L.

```
even_numbers([],[]).

even_numbers([H|T],L1):-

  integer(H),

  (

    H mod 2 =:= 0

    -> L1 = [H|T1], even_numbers(T,T1)
    ; even_numbers(T,L1)

    ).


odd_numbers([],[]).


odd_numbers([H|T],L1):-

  integer(H),

  (

    H mod 2 =:= 1

    -> L1 = [H|T1], odd_numbers(T,T1)

    ; odd_numbers(T,L1)

    ).


split_list:-

  write('Number of items for List? '),

  read(N),

  length(L, N),

  maplist(read, L),

  write('List L= '), write(L), nl,

  even_numbers(L, L1),

  write('Even Number List L1= '), write(L1), nl,

  odd_numbers(L, L2),

  write('Odd Number List L2= '), write(L2), nl.
```



```
?- consult('q14.pl').
true.

?- split_list.
Number of items for List? 4
|: .
|: 1.
|: 2.
|: 3.
|: 4.
List L= [1,2,3,4]
Even Number List L1= [2,4]
Odd Number List L2= [1,3]
true.
```