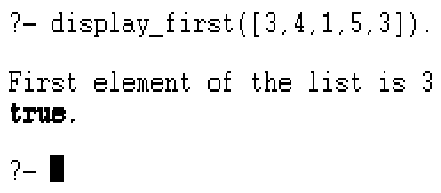
Practical – 5

Program 1: To display first element of a list.

```
display_first([H|_]):-
          nl, format('~s ~w', ['First element of the list is', H]).
```

Output :

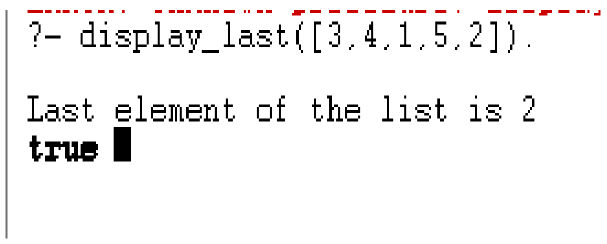


```
?- display_first([3,4,1,5,3]).

First element of the list is 3
true.

?- 
```

Program 2: To display last element of a list.

```
display_last([_|T]):-
   T \= [], display_last(T).

display_last([H|_]):-
   nl, format('~s ~w', ['Last element of the list is', H]).
```

Output:

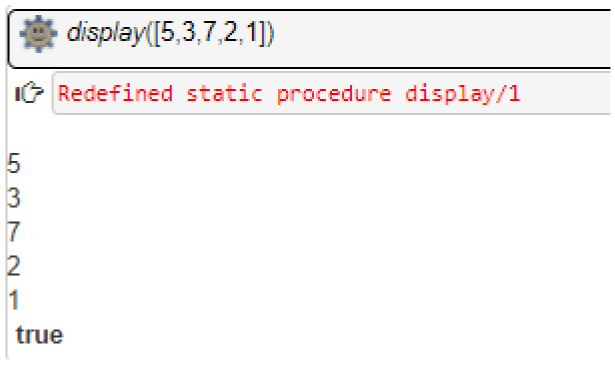


```
?- display_last([3,4,1,5,2]).

Last element of the list is 2
true 
```

Program – 3: To display all elements of a list

```
display([]).

display([H|T]):-
   nl, write(H), display(T).
```

Output:

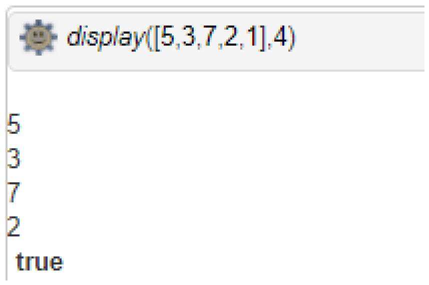


```
display([5,3,7,2,1])
Redefined static procedure display/1

5
3
7
2
1
true
```

Program 4: To display elements up to specified index of a list.

```
display(_, Index):-
   Index < 0, write('Negative index not allowed!').

display(List, Index):-
   display_elements(List, Index, 0).

display_elements([H|T], Index, C):-
   H \= [], C < Index ->
   (nl, write(H), display_elements(T, Index, C+1)); !.
```

Output:

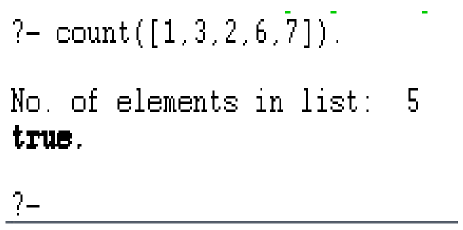


```
display([5,3,7,2,1],4)

5
3
7
2
true
```

Program 5: To count the number of elements in a list.

```
count(List):-
   count_next(List, 0).

count_next([], C):-
   nl, format('~s ~w', ['No. of elements in list: ', C]).

count_next([_|T], C):-
   C1 is C+1,
   count_next(T, C1).
```

Output:

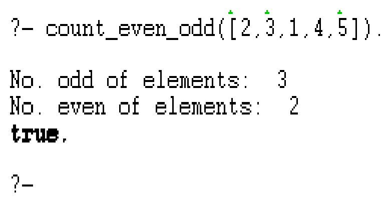


```
?- count([1,3,2,6,7]).

No. of elements in list:  5
true.

?-
```

Program 6: To count odd and even elements of a list.

```
count_even_odd(List):-
   count(List, 0, 0).

count([], Co, Ce):-
   nl, format('~s ~w', ['No. odd of elements: ', Co]),
   nl, format('~s ~w', ['No. even of elements: ', Ce]).

count([H|T], Co, Ce):-
   H mod 2 =:= 0 ->
   (Ce1 is Ce+1, count(T, Co, Ce1));
   (Co1 is Co+1, count(T, Co1, Ce)).
```

Output :



```
?- count_even_odd([2,3,1,4,5]).

No. odd of elements:  3
No. even of elements:  2
true.

?-
```