# CS354
# Assignment-2

**Name: Somya Mehta** **Roll No:190001058**

## Q1.

## Code:-

```
analyse_list(X) :-
        % if input is not list, fail.
        \+ is_list(X),!.
analyse_list([]) :-
        % if input is an empty list
        write('This is an empty list'),!.
analyse_list(X) :-
        %if list is valid
```

% output head and tail of the list

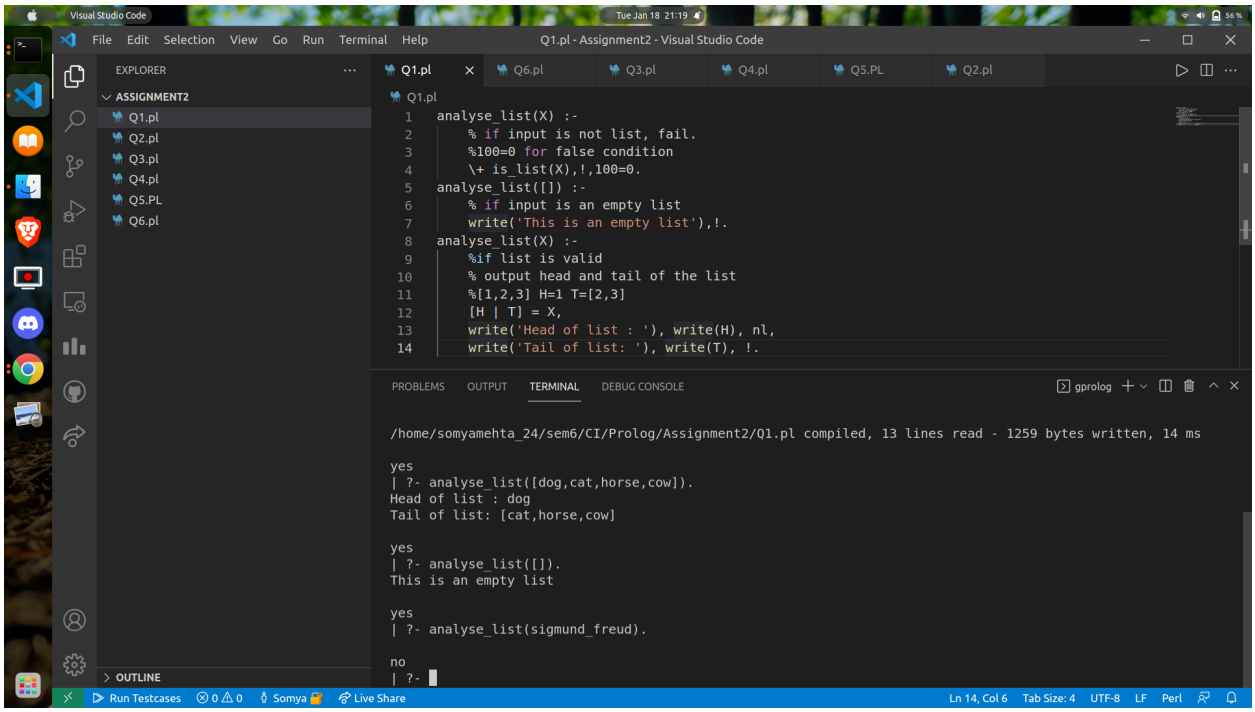
%[1,2,3] H=1 T=[2,3]

[H | T] = X,

write('Head of list : '), write(H), nl,

write('Tail of list: '), write(T), !.

## ScreenShot:-




## Q2.

## Code:-

remove_duplicates([], []). /* empty list as input */

```
remove_duplicates([H | T], L) :-
        /* if Head is a member of Tail */
        member(H, T),
        remove_duplicates(T, L) ,!. /* use ! to prevent backtracking */

remove_duplicates([H | T], [H | L]) :-
        % if Head is not a member of Tail => Add Head to List
        % and call remove_duplicates with Tail and updated List
        remove_duplicates(T, L).
```

## ScreenShot:-

```
| ?- ['Q2.pl'].
compiling /home/somyamehta_24/sem6/CI/Prolog/Assignment2/Q2.pl for byte code...
/home/somyamehta_24/sem6/CI/Prolog/Assignment2/Q2.pl compiled, 11 lines read - 963 bytes written, 5 ms

yes
| ?- remove_duplicates([1,2,3,11,1,1,1,1,2,3],X).

X = [11,1,2,3]

yes
```

## Q3.

## Code:-

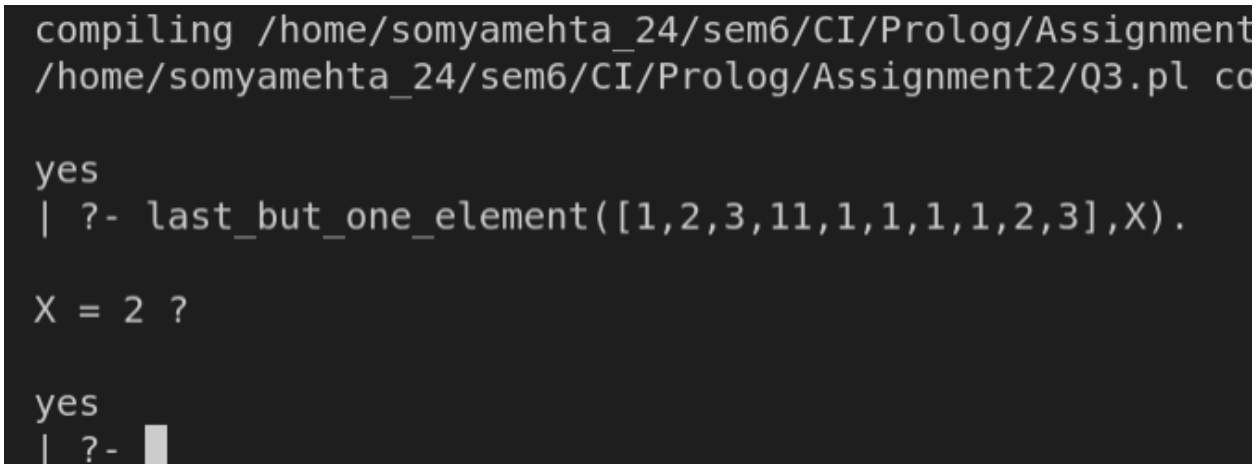
```
        % if only 2 elements are present in the list, assign first one to X
        last_but_one_element([X,_], X).
```

% otherwise recurse the function over tail of the list

last_but_one_element([_|T], X) :-

last_but_one_element(T, X).

## ScreenShot:-




## Q4.

## Code:-

%Base Case

%If K is 1 then our ans is the head of List

element_at(X,[X|_],1).

% If we have to find Kth element then we will remove first element

% from list then find (K-1)th element

element_at(X, [_|T], K) :-

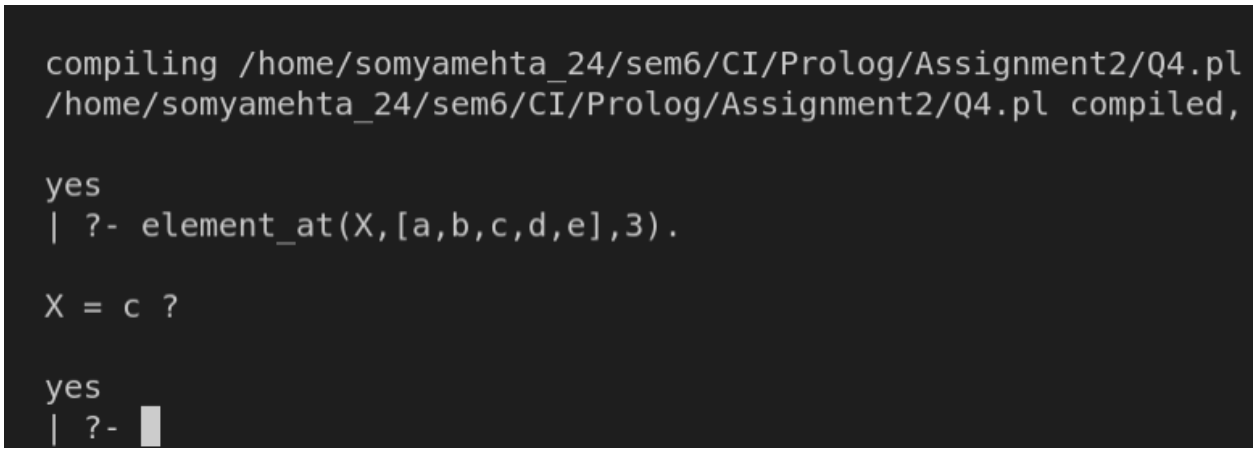
%Decrement value of K by 1

K > 1,

KK is K-1,

element_at(X, T, KK).

## ScreenShot:-




## Q5.

## Code:-

%Base Case

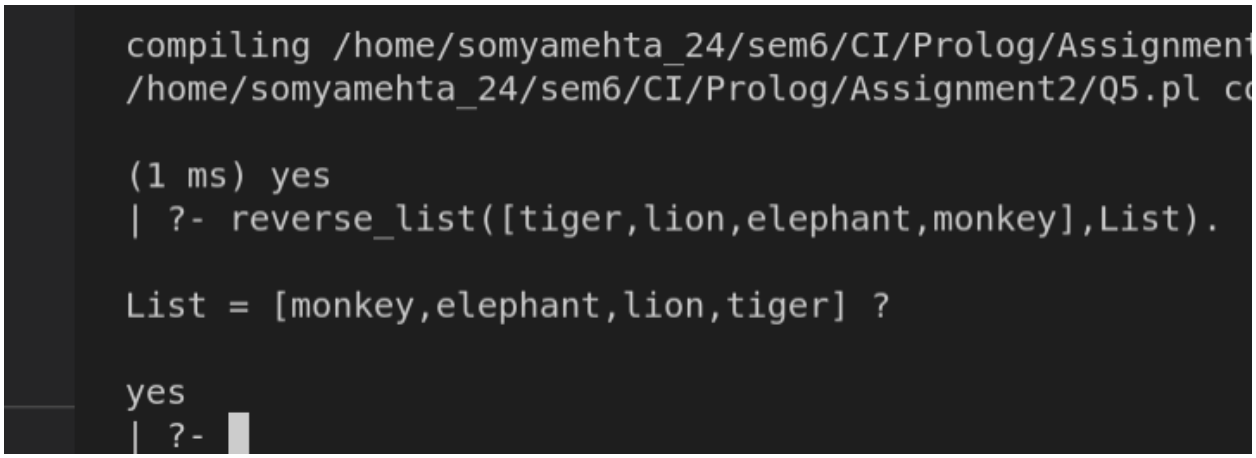
reverse_list([], []).% empty list then return true

%Single Element->Reversed list will be as it is

reverse_list([X], [X]).

```
reverse_list([H|T], X) :-
        %Reverse the Tail
        reverse_list(T, Temp),
        % add Temp to Head
        append(Temp, [H], X).
```

## ScreenShot:-




## Q6.

## Code:-.

```
check_palindrome([]).%empty list
%Base Case
%If list has single element then it is pallindrome
check_palindrome([_]).
%[H|T] in this H will be first element and [H] will be last element from
% append function
```

%if both are same then check for List T otherwise false

```
check_palindrome(X) :-
        append([H|T], [H], X),
        check_palindrome(T).
```

## ScreenShot:-