ASSIGNMENT - 2

ANIKEIT SETHI(190001003)

Q1)

```
analyse_list(L) :-
      L = [H | T] ,
      format('This is the head of your list:~w~n',[H]),
      format('This is the tail of your list:~w~n',[T]).

analyse_list([]) :-
     % input is an empty list
     write('This is an empty list'),!.


analyse_list(X) :-
     % if input is not list, prevent backtracking, fail.
      \+ is_list(X),!,1=0.
```

Q2.)

remove_duplicates([], []). % empty list input

remove_duplicates([H | T], Result) :-

```
% Head is a member of Tail => H is repeating => remove head from list
      member(H, T),!,
      remove_duplicates(T, Result) % use cut to prevent backtracking

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

Q3.)

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

%first case fails=>recurse the function over tail of the list
last_but_one([_|w], y) :
      last_but_one(W, y).
```

Q4.)

```
kth_element(X, [X|_], 1).

kth_element(X, [_|T], K) :- kth_element(X, T, Knew), Knew is K - 1.
```

Q5.)

```
rev([],[]).%Empty

rev([R],[R]).%single element

rev([H|T],R):- rev(T,TEMP), append(TEMP,[H],R).
```

Q6.)

```
pali([]). % empty list is palindrome

pali([_]). % single element list is also palindrome
```

```
pali(L) :-
      % H|Rem assigns H = first element, Rem = 2nd to last element
      % [H] => last element
      % if H from both the statements is same => Check if Rem is
      palindrome or not
      append([H|Rem], [H],
      L), pali(Rem).
```