Expt 2:

Sets are powerful data structure that can be natural is expressed using list in prolog. In prolog list elements are enclosed by brackets and separated by commas.

Ex: [1,2,3,4]
    [John, x]
    [(Mary,joe],[bob,carol,ted,alice])
    [A,[e,n,c];4]
    []

Another way to represent a list is to use the head or tail notation [H/T]

Here the head of the list H is separated from the tale of the list T by a vertical bar.

The tale of the list is the original list with the first element removed. The tale of the list is always a list even if its empty list.

In prolog the H/T notation is used together with unification to combine and breakup lists. For example, suppose we have the following list:

[bob,Carol, Ted]

Here the various matches we would obtain using H/T:

[X/Y],X=bob Y=[carol,ted]

[X,Y/Z],X=bob, Y=carol, Z=[ted], w=[]

Program

```
list_member(X,[X|_]).
list_member(X,[_|TAIL]) :- list_member(X,TAIL).
list_concat([],L,L).
list_concat([X1|L1],L2,[X1|L3]) :-
list_concat(L1,L2,L3).
delete(X,[X|T],T).
delete(X,[H|T],[H|NT]):-delete(X,T,NT).
```

Input &Output:

```
?- list_member(a,[a,b,c]).
true .
?- list_concat([1,2,3],[4,5,6],L).
L = [1, 2, 3, 4, 5, 6].
?- delete(6,[4,5,6],L).
L = [4, 5]
```

Expt 3:

Intersection: it will return those elements that are present both list. Let us define a clause called list_intersection(L1,lL2,L3), so this will take it L1 and L2 and perform intersection operation and store result into L3.

So consider L1=[a,b,c,d,e], L2=[a,e,i,o,u], then L3=[a,e]

Here we will use the list_member() clause to check if one element is present in a list or not.

Union: latest define a clause called list_union(L1,L2,L3). So this will take L1 and L2 and perform union on them and store the result in L3. As you know it too list have the same element twice then after union there will be only one so we need another helper cause to check the membership.

Program

```
list_member(X,[X|_]).
list_member(X,[_|TAIL]) :- list_member(X,TAIL).

list_union([X|Y],Z,W) :-
list_member(X,Z),list_union(Y,Z,W).
list_union([X|Y],Z,[X|W]) :- \+ list_member(X,Z),
list_union(Y,Z,W).
list_union([],Z,Z).
list_member(X,[X|_]).
list_member(X,[_|TAIL]) :- list_member(X,TAIL).

list_intersect([X|Y],Z,[X|W]) :-
  list_member(X,Z), list_intersect(Y,Z,W).
list_intersect([X|Y],Z,W) :-
  \+ list_member(X,Z), list_intersect(Y,Z,W).
list_intersect([],Z,[]).
```

Input &Output:

```
?- list_intersect([1,2,3,4],[1,a,b,4],X).
X = [1, 4] .

?- list_intersect([1,2,3,4],[1,a,b,5],X).
X = [1] .
?- list_union([1,2,3,4],[1,a,b,4],A).
A = [2, 3, 1, a, b, 4] .

?- list_union([1,2,3,4],[a,b],A).
A = [1, 2, 3, 4, a, b].
```

EXPT 4:
In prologue a minute even program can be implemented using rules and predictions.
1. MENU Options: define menu options as fact or rules. Each option corresponds to a predicate that performs a specific task or computation.
2. Displaying the MENU: create a predicate eg 'disp_menu/0' , which is responsible for displaying menu options to the user. Inside predicate use 'write/1' predicate to print the menu options on the console.
3. Handling user input: Eg: 'process_choice/1' takes user's input as an argument. Inside predicate use combination of rules, conditional statements to handle the chosen option. Match the input choice with the defined menu options using unification.
4. Starting the program: 'start_prog' which serves as the entry point to your program. 'display_menu' predicate to initiate the menu display & user interaction.

Program

```
def menu():
    while True:
        choice = int(input(
            """
            1. Member
            2. Concatenation
            3. Add
            4. Delete
            5. Permutation
            6. Exit

            Enter your choice: """
        ))
        if choice == 1:
            element = input("Enter element: ")
            set = input("Enter set: ").split()
            result = element in set
            print(f"{element} {'is a member' if result
else 'is not a member'} of {set}")
        elif choice == 2:
            list1 = input("Enter first list: ").split()
            list2 = input("Enter second list: ").split()
            result = list1 + list2
            print(f"Concatenated list: {result}")
        elif choice == 3:
            element = input("Enter element to add: ")
            list = input("Enter list: ").split()
            result = list.append(element)
            print(f"Updated list: {list}")
        elif choice == 4:
            element = input("Enter element to delete: ")
            list = input("Enter list: ").split()
            try:
                result = list.remove(element)
                print(f"Element '{element}' deleted.")
            except ValueError:
                print(f"Element '{element}' not found in
list")
            print(f"Updated list: {list}")
        elif choice == 5:
            list = input("Enter list: ").split()
            from itertools import permutations
            result = list(permutations(list))
            print(f"Permutations: {result}")
        elif choice == 6:
            break
        else:
            print("Invalid choice. Please try again.")

menu()
```