Question 1

- Write a program that eliminates consecutive duplicates of a number
- Sample goal: eliminate_dupli ([1,1,1,2,2,2,2,2,3,3,4] , L).
- L=[1,2,3,4]

```
Domains
    lst = Integer*
Predicates
    nondeterm eliminate_duplicates(lst, lst)
Clauses
    eliminate_duplicates([], []).
    eliminate_duplicates([X], [X]).
    % If the first two elements are the same, skip the first one
    eliminate_duplicates([H,H|T], L) :- eliminate_duplicates([H|T], L).
    % If the first two elements are different, include the first one
    eliminate_duplicates([H,X|T], [H|T1]) :- H <> X, eliminate_duplicates([X|T],
T1).
Goal
    eliminate_duplicates([1,1,1,2,2,2,2,2,3,3,4], L).
```

Question 2

- Write a program that duplicates the elements of a list a given number of times.
- Sample goal: duplicate([a,b,c] , 3)
- L=[a,a,a,b,b,b,c,c,c]

Solving using append:

```
Domains
    lst = Symbol*
Predicates
    nondeterm append(lst, lst, lst)
    nondeterm duplicate_element(Symbol, Integer, lst).
    nondeterm duplicate(lst, Integer, lst)
Clauses
    append([], L, L).
    append([H|T], L, [H|T1]) :- append(T, L, T1).

    duplicate_element(_, 0, []).
    duplicate_element(X, N, [X|L]) :-
        N > 0,
        N1 = N - 1,
        duplicate_element(X, N1, L).

    duplicate([], _, []).
    duplicate([H|T], N, L) :-
        duplicate_element(H, N, DE),
        duplicate(T, N, RD),
```

```
        append(DE, RD, L).
Goal
    duplicate([a,b,c], 3, L).
```

Question 4

- Write a program that calculates the dot products of two vectors
- Sample goal : dot([1,2,3] , [4,5,6] , D)
- D=32

```
Domains
    lst = Integer*
Predicates
    nondeterm dot(lst, lst, Integer)
Clauses
    dot([], [], 0).
    dot([H1|T1], [H2|T2], D) :-
        dot(T1, T2, D1),
        D = H1 * H2 + D1.
Goal
    dot([1,2,3], [4,5,6], D).
```

Question 5

- Write a program that calculates the product of two sets
- Sample goal : product([a,b],[c,d,f],L)
- L=[(a,c),(a,d),(a,f),(b,c),(b,d),(b,f)]

```
Domains
    lst = Symbol*
    pair = lst*
Predicates
    nondeterm product(lst, lst, pair)
    nondeterm product_element(Symbol, lst, pair)
    nondeterm append(pair, pair, pair)


Clauses
    append([], L, L).
    append([H|T], L, [H|T1]) :- append(T, L, T1).

    product([], _, []).
    product([H|T], L, P) :-
        product_element(H, L, PE),
        product(T, L, P1),
        append(PE, P1, P).

    product_element(_, [], []).
```

```
    product_element(X, [H|T], [[X, H]|PE]) :-
        product_element(X, T, PE).
Goal
    product([a,b], [c,d,f], L).
```