Vivek Poojari
2019130050
TE Comps Batch – C

# Experiment-6

**Aim:** To solve problems using Prolog Programming and solve the given 5 tasks that are list operations and Huffman coding.

**Problem statements:**

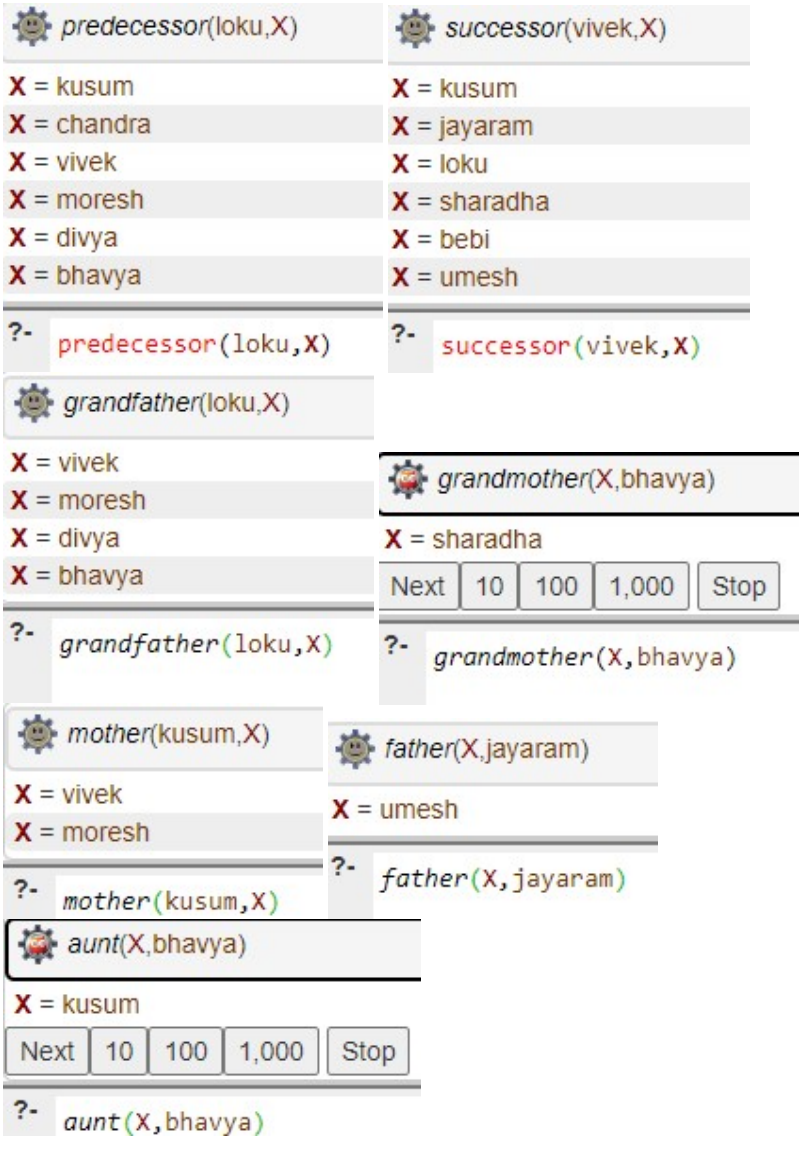
1) Create a family tree using PROLOG. It should have rules for father, mother, brother, sister, grandparent, uncle, aunt, predecessors, successors.

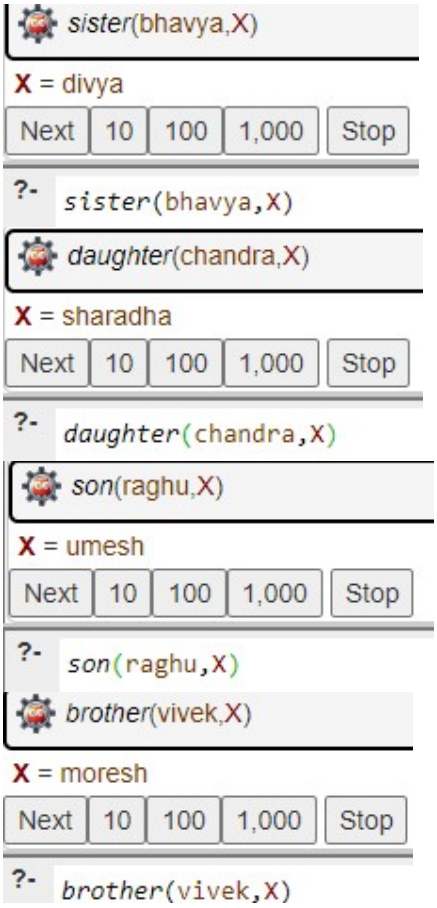
   **Code:**

```
parent(bebi,jayaram).
parent(umesh,jayaram).
parent(umesh,raghu).
parent(bebi,raghu).
parent(kusum, vivek).
parent(jayaram, vivek).
parent(chandra, divya).
parent(shekhar, divya).
parent(loku, kusum).
parent(sharadha, kusum).
parent(sharadha, chandra).
parent(loku, chandra).
parent(chandra, bhavya).
parent(shekhar, bhavya).
parent(kusum, moresh).
parent(jayaram, moresh).
female(bebi).
female(kusum).
female(bhavya).
female(divya).
female(chandra).
female(sharadha).
male(shekhar).
male(raghu).
male(jayaram).
male(moresh).
male(vivek).
male(loku).
```

```
male(umesh).
%rules
predecessor(X, Y) :- parent(X, Y).
predecessor(X, Y) :- parent(X, A),predecessor(A, Y).
successor(X, Y):- son(X,Y).
successor(X, Y):- daughter(X, Y).
successor(X, Y):- son(X, A), successor(A, Y).
successor(X, Y):- daughter(X, A), successor(A, Y).
grandfather(X, Y):- parent(X, A), parent(A, Y), male(X).
grandmother(X, Y):- parent(X, A), parent(A, Y), female(X).
mother(X, Y):- parent(X, Y), female(X).
father(X, Y):- parent(X, Y), male(X).
aunt(X, Y):- sister(X, Z), parent(Z, Y).
uncle(X, Y):- brother(X, Z), parent(Z, Y).
sister(X, Y):- parent(A, X), parent(A, Y), female(X), X \= Y.
brother(X, Y):- parent(A, X), parent(A, Y), male(X), X \= Y.
son(X, Y):- parent(Y, X), male(X).
daughter(X, Y):- parent(Y, X), female(X).
```

**Output:**






Given a list [a,a,a,a,b,b,b,c,c] write a function that does the following rle([a,a,a,a,b,b,b,c,c],X) X: [a,b,c]

**Code:**

```
%stopping condition
remove_duplicates([X],[X]).

%if next element is same
remove_duplicates([H, H|T],X) :-
    remove_duplicates([H|T],X).

%if next element not same
remove_duplicates([H, P|T],[H|Y]) :-
    H\=P,
```

remove_duplicates([P|T],Y).


3) Given a list [a,b,c,d,e,f,g] write a function that does the following slice([a,b,c,d,e,f,g],2,5,X) X: [c,d,e,f]

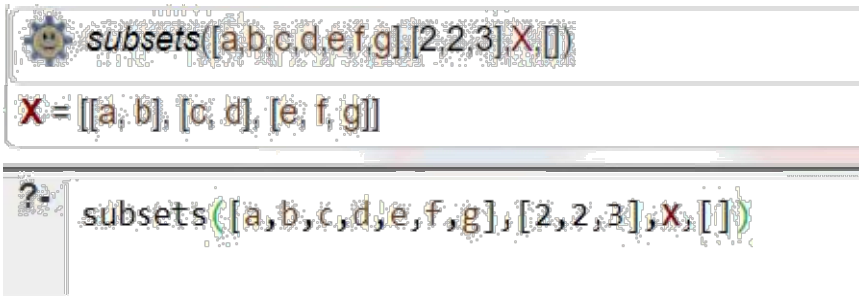
**Code:**

```
% puts last value of slice in output
list slice([X|_], 0, 0, [X]).

% called until last slice index is reached
slice([X|T], 0, L, [X|T_new]) :- L >0, L_new is L - 1, slice(T, 0, L_new, T_new).

% called until initial slice index is reached
slice([_|T], F, L, Output) :- F > 0, F_new is F - 1, L_new is L - 1, slice(T, F_new,
L_new, Output).
```


4) Group list into sublists according to the distribution given For example subsets([a,b,c,d,e,f,g],[2,2,3],X,[]) should return X = [[a,b][c,d][e,f,g]] The order of the list does not matter

**Code:**

```
el(X,[X|L],L).
el(X,[_|L],R) :-
   el(X,L,R).
selectN(0,_,[]) :- !.
selectN(N,L,[X|S]) :-
   N>0,
   el(X,L,R),
   N1 is N-1,
   selectN(N1,R,S).
subsets([],[],[],[]).
subsets(G,[N1|Ns],[G1|Gs],[]) :-
   selectN(N1,G,G1),
   subtract(G,G1,R),
    subsets(R,Ns,Gs,[]).
```

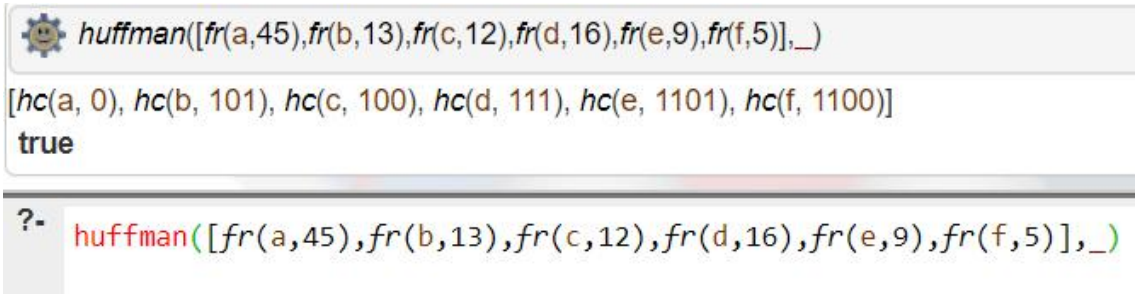



5) Huffman Code We suppose a set of symbols with their frequencies, given as a list of fr(S,F) terms. Example: [fr(a,45),fr(b,13),fr(c,12),fr(d,16),fr(e,9),fr(f,5)].
Our objective is to construct a list hc(S,C) terms, where C is the Huffman code word for the symbol S. In our example, the result could be Hs =[hc(a,'0'), hc(b,'101'), hc(c,'100'), hc(d,'111'), hc(e,'1101'), hc(f,'1100')].
The task shall be performed by the predicate huffman/2 defined as follows: % huffman(Fs,Hs) :- Hs is the Huffman code table for the frequency table Fs

**Code:**

```
huffman(Fs,Cs) :-
   initialize(Fs,Ns),
   make_tree(Ns,T),
   traverse_tree(T,Cs).

initialize(Fs,Ns) :- init(Fs,NsU), sort(NsU,Ns).

init([],[]).
init([fr(S,F)|Fs],[n(F,S)|Ns]) :- init(Fs,Ns).

make_tree([T],T).
make_tree([n(F1,X1),n(F2,X2)|Ns],T) :-F is
   F1+F2,
   insert(n(F,s(n(F1,X1),n(F2,X2))),Ns,NsR),
   make_tree(NsR,T).

insert(N,[],[N]) :- !.
insert(n(F,X),[n(F0,Y)|Ns],[n(F,X),n(F0,Y)|Ns]) :- F < F0, !.
insert(n(F,X),[n(F0,Y)|Ns],[n(F0,Y)|Ns1]) :- F >= F0, insert(n(F,X),Ns,Ns1).

traverse_tree(T,Cs) :-
    traverse_tree(T,'',Cs1-[]), sort(Cs1,Cs),
    write(Cs).

traverse_tree(n(_,A),Code,[hc(A,Code)|Cs]-Cs)
    :-atom(A).

traverse_tree(n(_,s(Left,Right)),Code,Cs1-Cs3)
   :-atom_concat(Code,'0',CodeLeft),
   atom_concat(Code,'1',CodeRight),
```

```
traverse_tree(Left,CodeLeft,Cs1-Cs2),
traverse_tree(Right,CodeRight,Cs2-Cs3).
```




**Conclusion:**

The goal of this experiment was to learn prologue programming and complete five tasks including list operations such as duplicate removal, slicing, subsets, and huffman coding, as well as generating a family tree and answering a few questions. The Huffman coding algorithm is a lossless data compression method. The concept is to give variable-length codes to input characters, the lengths of which are determined by the frequency of the related characters. The smallest code is assigned to the most frequently occurring character, whereas the largest code is assigned to the least frequently occurring character. Rules, facts, and queries are the three basic constructs of Prolog. Prolog programs are essentially knowledge bases, consisting of rules, facts, and questions, the answers to which are sought from this knowledge base and sent as a boolean value. The language Prolog is utilised in AI because it facilitates the administration of recursive techniques and pattern matching.