# Logic Programming

## Assignment 6

**Due: Next Section.**

Note: This task is individual task, The lab Instructor should evaluate a running program for it.

---

**1. (**5 Points**) Using difference lists, write a predicate flatten_list that flattens nested occurrences of lists. Sample runs:**

?- flatten_list([a,b,[1,2,3,[z]]], X).
X = [a,b,1,2,3,z]

?- flatten_list([a,b,c], X).
X =[a,b,c]

**2. (**5 Points**) Write a ternary predicate delete_all(item,list,result) that is true if result is obtained from 'list' by deleting all occurrences of 'item'. Use cut to prevent backtracking. Sample run:**

?-delete_all(a,[a,b,c,a,d,a],X).
X=[b,c,d];
false
?-delete_all(a,[b,c,d],X).
X=[b,c,d];
false

**3. (**10 points**) The program below is supposed to compute prime factorization of a given positive integer. Its code is badly formatted. Read Coding Guidelines for Prolog (http://arxiv.org/pdf/0911.2899) and reformat the program, including the comments, according to them.**

```
% Prime factorization of a given positive integer number. prime_factors(N, L)
succeeds if the L is the list of prime factors of N.

prime_factors(N,L):- N >0,prime_factors(N,2,L).

% prime_factors(N,F,L) succeeds if L is the list of prime factors of N such that N
does not have a prime factor less than K.

prime_factors(1,F,[]) :- !.
prime_factors(N,F,[F|L]) :- % N is multiple of F
                            R is N // F, N =:= R * F, !,
                            prime_factors(R,F,L).
```

```
prime_factors(N,F,L) :-
next_factor(  N,F,NF),
prime_factors(N,NF,L).        % N is not multiple of F
% next_factor(N,F,NF) succeeds if F does not divide N and NF is the next larger
candidate to be a factor of N.

next_factor(N,2,3):- !.
next_factor(N,F,NF) :-F *F < N,!,
NF is F + 2.  next_factor(N,_,N).                                    % F > sqrt(N)
```