**Practical 10**

1. Define a predicate nu/2 ("not unifiable") which takes two terms as arguments and succeeds if the two terms do not unify. For example:

nu(foo,foo).

no

nu (foo,blob).

yes

nu(foo,X).

no

You should define this predicate in three different ways:

1. First (and easiest) write it with the help of = and \+.

nu(X,Y) :- \+ X=Y.

1. Second write it with the help of =, but don't use \+.

nu(X,Y) :- ( X=Y -> fail ; true ).

1. Third, write it using a cut-fail combination. Don't use = and don't use \+.

nu(X,X) :- !,fail.

nu(\_,\_) :- !.

1. Define a predicate unifiable(List1,Term,List2) where List2 is the list of all members of List1 that match Term, but are not instantiated by the matching. For example,

unifiable([X,b,t(Y)],t(a),List~~]~~).

should yield

List = [X,t(Y)].

Note that X and Y are still not instantiated. So the tricky part is: how do we check that they match with t(a) without instantiating them? (Hint: consider using the test \+ (term1 = term2). Why? Think about it. You might also like to think about the test \+(\+ (term1 = term2)).)

unifiable([],\_,[]).

unifiable([X|Xs],Term,[X|Result]) :-

\+(\+ X=Term),

unifiable(Xs,Term,Result).

unifiable([X|Xs],Term,Result) :-

\+ X=Term,

unifiable(Xs,Term,Result).