Download the accompanying zip file from Blackboard. Solve each problem below using Prolog on the cs-parallel server. Do not modify the provided predicate signatures or file names. Test your predicates using the examples in the provided zip file; you are encouraged to also create some additional examples to test more thoroughly. When you are ready to submit, compress your solution files into a zip file, and upload to Blackboard. Double-check that you have submitted all the files you intended.

1. Suppose a Prolog database defines family relations of the form parent(X, Y), which means that X is a parent of Y. Define this new predicate: half_cousin(X, Y). Half-cousins share only one grandparent.

2. Suppose a Prolog database defines family relations of the form parent(X, Y), which means that X is a parent of Y. Define this new predicate: double_cousin(X, Y). Double-cousins share both sets of grandparents.

3. setcover(N, L, R) takes integer N and list L whose nested lists are subsets of a universal set of elements. It returns list R which is a subsequence of L having length ≤ N and whose union is the same universal set of elements. Example:
   setcover(3, [[a,b],[a,c],[a,d],[a,e],[c,e]], R) yields R = [[a,b],[a,d],[c,e]] whose length is 3 and whose union is the universal set [a,b,c,d,e].
   You may use the following helper predicates for set union and set difference if you wish:

```
union([ ],X,X).
union([H|T],X,U) :- member(H,X), !, union(T,X,U).
union([H|T],X,[H|U]) :- union(T,X,U).
diff([ ],_,[ ]).
diff([H|T],X,D) :- member(H,X), !, diff(T,X,D).
diff([H|T],X,[H|D]) :- diff(T,X,D).
```

4. reject(P, L, R) removes all the elements of L that satisfy predicate P, and returns the result in R. Example: reject(number, [a,5,b,3.14,c,89,[ ],2.7,d,[6]] ,R) yields R = [a,b,c,[ ],d,[6]].

5. multimap(F, L, R) takes a predicate F and a list L with nested sublists. It applies F to each list that consists of corresponding elements of all the sublists, and returns a list of the results in R. For full credit, you should handle the general case when the sublists of L might have different lengths. Examples:
   sum([ ], 0).
   sum([H|T], R) :- sum(T,X), R is H+X.
   multimap(sum, [[1,2,3,4],[5,6,7],[8,9,10,11],[12,13,14]], R) returns R = [26,30,34].
   prod([ ], 1).
   prod([H|T], R) :- prod(T,X), R is H*X.
   multimap(prod, [[2,3,4],[5,6,7,8],[9,10,11]] , R) returns R = [90,180,308].

6. bools(L) enforces that each element of list L is either true or false. solve(E, B) determines values for the variables in logical expression E so that E will evaluate to Boolean value B. Examples: bools([X,Y]), solve(and(or(X,Y),not(X)), true) returns X=false, Y=true
   bools([X,Y,Z]), solve(or(not(and(X,Y)),Z), false) returns X=true, Y=true, Z=false.

7. When working with difference lists, you must be careful not to accidentally create an infinite list. The predefined predicate unify_with_occurs_check(X,Y) is useful for preventing infinite lists; it behaves similarly to X=Y except it will not unify any variable with a compound expression containing that variable. Example: Suppose we want to write predicate null(X-Y) so that it will succeed if and only if the difference list X-Y represents an empty list.

   - Here is an incorrect definition: null(X-Y) :- X=Y. It is correct when X and Y are bound to finite lists, but unfortunately the query null([a,b,c,d|T]-T) returns true because variable T unifies with [a,b,c,d|T] and yields an infinite list T = [a,b,c,d,a,b,c,d,a,b,c,d,...].
   - Here is a correct definition: null(X-Y) :- unify_with_occurs_check(X,Y). Now the query null([a,b,c,d|T]-T) returns false, because variable T does not unify with [a,b,c,d|T].

   Write these predicates that use difference lists:

   a. dlength(A-B, N) takes a difference list A-B, and returns the number of elements that are in A but not in B. Example: dlength([a,b,c,d|T]-T, N) yields N=4.

   b. dreverse(A-B, C-D) succeeds if difference list C-D is the reverse of difference list A-B. Example: dreverse([a,b,c,d|T]-T, X-Y) yields X = [d,c,b,a|Y].

   c. drotateleft(A-B, K, C-D) succeeds if difference list C-D is obtained by rotating the elements of difference list A-B by K positions to the left. Example:
   drotateleft([a,b,c,d,e,f|T]-T, 2, X-Y) yields X = [c,d,e,f,a,b|Y].

   d. drotateright(A-B, K, C-D) succeeds if difference list C-D is obtained by rotating the elements of difference list A-B by K positions to the right. Example:
   drotateright([a,b,c,d,e,f|T]-T, 2, X-Y) yields X = [e,f,a,b,c,d|Y].