Prolog Basic
Example Test Questions

1. Write a Prolog predicate \texttt{split(TheList, Evens, Odds)} that asserts the following
   • \texttt{Odds} contains all the items in the odd positions of \texttt{TheList} that are not the empty list
   • \texttt{Evens} contains all the items in the even positions of \texttt{TheList}, including the empty list

   Use = and \_\_\_ to distinguish cases.

2. Define the predicate \texttt{odd_list(a_list)} where \texttt{a_list} is a list of atoms. The predicate asserts the list
   contains an odd number of elements. Do NOT USE the length predicate or numbers.

   \begin{verbatim}
   ?- odd_list([]). No
   ?- odd_list([one]). Yes.
   ?- odd_list([one, two]). No.
   ?- odd_list([one, two, three]). Yes.
   ?- odd_list(one). No.
   \end{verbatim}

3. A. Consider the following predicate, \texttt{mystery(List1,List2,Result)}.
   \begin{verbatim}
   mystery([],L2,L2).
   mystery([|,L1) :- L1 = [___].
   mystery([H1|T1],[H2|T2],[H1|T3]) :- mystery(T1,[H2|T2],T3).
   mystery([H1|T1],[H2|T2],[H2|T3]) :- mystery([H1|T1],T2,T3).
   \end{verbatim}
   What does the following query produce, if semi-colon, ;, is used to find more than one answer.
   \texttt{mystery([1,2],[a,b],R)}.

   B. Give a brief English description of the predicate \texttt{mystery}.

   C. Replace the last two rules of the predicate \texttt{mystery} with a single rule.

4. E What is the difference between the following two rules?
   \begin{verbatim}
   blah :- a(X) , b(X).
   blah :- a(_) , b(_).
   \end{verbatim}

5. Write a definition of the predicate \texttt{removeNil (List, Rlist)} that asserts \texttt{Rlist} is the same as \texttt{List}
   except that all instances of the item [\_] (the empty list) have been removed at all levels of \texttt{List}.

6. Define a Prolog predicate \texttt{remove2nd(List, NewList)} that asserts that \texttt{NewList} is the same a \texttt{List}
   except that the second top-level item is removed.
   \begin{verbatim}
   remove2nd([a,b,c], [a,c]) \rightarrow yes
   remove2nd([a,b], X) \rightarrow X = [a]
   remove2nd ([a], X) \rightarrow X = [a]
   \end{verbatim}
7. Write a Prolog predicate facti (N, F) that asserts F is the factorial of N. Ensure that it works for the query facti (N, F).

8. Define a Prolog predicate flatten (List, FlattenedList) that asserts List is any nested list of atoms and FlattenedList is the same list with the nesting removed. The atom [] should also be removed. Your predicate should only produce one answer. You may use the built-in predicates \+ (not), ! and append. Do not use a helper predicate.

? flatten([ [ a, [ b, c ], d ], [ [ e ] ], [ f ] ], X).
X = [ a, b, c, d, e, f ];
no

? flatten([ [ a, [ ] ], [ [ c, d ], e ] ], X).
X = [ a, c, d, e ];
no

9. Write a prolog predicate insert_nth(item, n, into_list, result) that asserts that result is the list into_list with item inserted as the n’th element into every list at all levels. Counting begins at 1.

10. Write a predicate nth(N, Alist, Elem) such that Elem is the N’th item in the list Alist. nth(1, Alist, Elem) is true for the first item in the list.

11. Write a predicate index(Array, [I1, I2, ..., In], Elem) such that Array[I1, I2, ..., In] = Elem. There is no fixed size for the number of dimensions. You may use the predicate nth from part A if you wish but you do not have to. Assume index value 1 is the first item in the corresponding dimension.

12. Assume the prolog predicate gt (A, B) is true when A is greater than B. Use this predicate to define the predicate addLeaf(Tree, X, NewTree) which is true if NewTree is the Tree produced by adding the item X in a leaf node. Tree and NewTree are binary search trees. The empty tree is represented by the atom nil.

13. Write a Prolog predicate to remove the N’th item from a list.

14. A The predicate maximum(X, Y, M) is true if and only if M is the maximum integer of X and Y. The following is a variation of the definition that was discussed in class.

maximum(X, Y, M) :- X >= Y , M = X ; Y >= X , M = Y.

Is the predicate correct? Are there any circumstances when it may fail to give the expected answer? If it does fail, correct the definition in the simplest possible way.

15. Define a Prolog predicate sort(X, Y) that asserts that X is a list of integers and Y is the same list but sorted in ascending order. Your algorithm MUST be the following

Repeatedly choose the smallest remaining element from X and add it to Y.

Hint: use a helper predicate called smallest.
16.
Write a Prolog predicate, remove-nth(Before, After) that asserts the After list is the
Before list with the removal of every n’th item from every list at all levels. Counting begins at 1.
Precondition: N ≥ 1, Before and After are lists.

17.
A Define a predicate listCount(AList ,Count) that is true if AList contains Count
number of elements that are lists at the upper level. Define without using an accumulator. Use
"not" as defined in utilities.pro, to make similar cases unique, or else you may get more than one
count as an answer.
Examples:
listCount([b,a],N).
N = 0 ;
no

listCount([b,[a,[a],c],a], 1).
N = 0 ;
N = 1 ;
no

listCount([b,[a,[a],c],a, []],N).
N = 2 ;
no