

Prolog Basic Example Test Questions

1.

Write a Prolog predicate **split(TheList, Evens, Odds)** that asserts the following

- **Odds** contains all the items in the odd positions of **TheList** that are not the empty list
- **Evens** contains all the items in the even positions of **TheList**, including the empty list

Use = and \= to distinguish cases.

2.

Define the predicate **odd_list(a_list)** where 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.

```
?- odd_list([]).
No
?- odd_list([one]).
Yes.
?- odd_list([one, two]).
No.
?- odd_list([one, two, three]).
Yes.
?- odd_list(one).
No.
```

3.

A. Consider the following predicate, **mystery(List1, List2, Result)**.

```
mystery([], L2, L2).
mystery(L1, [], 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).
```

What does the following query produce, if semi-colon, ;, is used to find more than one answer.

```
mystery([1,2], [a,b], R).
```

B. Give a brief English description of the predicate **mystery**.

C. Replace the last two rules of the predicate **mystery** with a single rule.

4.

E What is the difference between the following two rules?

```
blah :- a(X) , b(X).
blah :- a(_) , b(_).
```

5.

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

6.

Define a Prolog predicate **remove2nd(List, NewList)** that asserts that **NewList** is the same a **List** except that the second top-level item is removed.

```
remove2nd([a,b,c], [a,c]) → yes
remove2nd([a,b], X) → X = [a]
remove2nd([a], X) → X = [a]
```

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.

Consider the following Prolog code.

```
pair(bert, ernie).
pair(bigbird, cookiemonster).
pair(bert, elmo).
pair(cookiemonster, tina).
pair(tina, bert).
single(oscar).
```

- A. What is the first result of the query `pair(A,B), pair(B,C), not(single(A))`.
 B. What is the first result of the query `pair(A,B), pair(B,C), !, not(single(A))`.
 C. What is the first result of the query `not(single(A)), pair(A,B), pair(B,C)`.

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.

Precondition: $n \geq 1$ and $n \leq 1 + \text{length}(\text{shortest list at any level in list})$

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: $N \geq 1$, `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).      listCount([b,[a,[a],c],a], 1).
N = 0 ;                 N = 1 ;
no                       no
```

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