2015 February 17

Prolog Accumulator
Example Exam Questions

For hole questions, replace accumulator in the following questions.

1.

Describe a general template for a Prolog predicate that counts with an accumulator.

2.

Describe a general template for a Prolog predicate that counts without using an accumulator.

3.

Describe the difference between Prolog predicates that count with an accumulator and those that count without an accumulator.

4.

A binary tree is defined by the compound term **bt(L, D, R)** and the empty tree is represented by a variable. The predicate **collect(Tree, Data)** asserts that **Data** is the collection of all the data items the **Tree**. Use an accumulator to collect the data, and use cut to eliminate useless choices. There are three possible orders pre-order, in-order, and post-order.

5.

Write a prolog predicate **count_atoms(L, C)** that asserts **C** is the number of atoms at all levels of a list **L**. The definition must use an accumulator. The predicate **atom(X)** asserts that **X** is an atom. Use cut to eliminate useless searching.
The Prolog predicate **allAtoms(List, Atoms)** asserts that **Atoms** is the list of all atoms at all levels of the list **List**, including duplicates. Recall that the predicate **atom(X)** asserts that X is an atom.

6.

The Prolog predicate **allAtoms(List, Atoms)** asserts that **Atoms** is the list of all atoms at all levels of the list **List**, including duplicates. Recall that the predicate **atom(X)** asserts that X is an atom.

A Give a definition of **allAtoms** that uses an accumulator to collect the atoms. Do not use append.
B Give a definition of **allAtoms** that uses difference lists (holes) to collect the atoms. Do not use append.

Variations: Collect all (1) numbers, (2) compound; terms with a given functor,

7.

Define a Prolog predicate **flatten (List, FlattenedList)** that asserts if **List** consists of any nested list of atoms then **FlattenedList** is the same list with the nesting removed. The atom [ ] should also be removed. Your predicate should only produce one answer. Use = and \= to distinguish cases. Do not use the predicates \+ (not), !. The fewer predicates you use the better.

```
?- flatten([ a, [ [ b, c ], d ], [ [ e ] ], [ f ] ] ], X).
X = [ a, b, c, d, e, f] ;
false
```

```
?- flatten([ a, [ [ ] ], [ [c. d] , e ] ], X).
X = [ a, c, d, e] ;
false
```

A. Complete a definition of **flatten** that uses an accumulator.

B. Complete a definition of **flatten** that uses a hole.

8.

Define a predicate **add_up_list(L, K)** that asserts that L and K are lists of integers where every element in K is the sum of all the elements in L up to the same position.

Precondition: L is an instantiated variable

Examples:

```
?- add_up_list( [1], K).
K = [1];
no

?- add_up_list( [1, 2], K).
K=[1,3] ;
no

?- add_up_list([ 1, 3, 5, 7], K).
K=[1, 4, 9, 16];
no
```

9.

Define a predicate `memCount(AList, Blist, Count)` that is true if `Alist` occurs `Count` times within `Blist`. You must use an accumulator. Use = and \= to distinguish cases.

Examples:

```
memCount(a,[b,a],N).
N = 1 ;
no

memCount(a,[b,[a,a,[a],c],a],N).
N = 4 ;
no

memCount([a],[b,[a,a,[a],c],a],N).
N = 1 ;
no
```

10.

Write a Prolog predicate `countLists(Alist, Ne, Nl)` that asserts `Nl` is the number of non-empty lists at the top level of `Alist` and `Ne` is the number of empty lists at the top level of `Alist`. Do not use accumulators in counting. Use = and \= to distinguish cases.

11.

Write a predicate `countBT(Tree, Count)` to count the number of nodes in a binary tree. Use an accumulator. `TREE` has the structure `bt(data, leftTree, rightTree)`. The empty tree is represented by an uninstantiated variable. The predicate `var(X)` returns true if `X` is an uninstantiated variable and false otherwise. Use `cut` or `not` to eliminate multiple counts.
Variations:
- Count the number of nodes that have one child
- Count the number of nodes that have two children.
- Count the number of leaf nodes in a tree.

12.

Write a Prolog predicate `countLR(Tree, CountLeft, CountRight)` that is true if an only if CountLeft is the number of nodes in `Tree` with only a left child and CountRight is the number of nodes in `Tree` with only a right child. The `Tree` has the structure

`bt(data, leftTree, rightTree)`.

The empty tree is represented by an uninstantiated variable. The predicate `var(X)` returns true if `X` is an uninstantiated variable and false otherwise. Use `cut` or `not` to eliminate multiple counts. You must use the accumulator method. You must document your solution.

13.

Write a predicate `diagOf(theMatrix,theDiag)` where theMatrix is a square matrix and theDiag is the diagonal of the matrix.