2009 November 30

Accumulator
Example Test 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.

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.

5.

Write a Prolog predicate **countBT(Tree, Count)** to count the number of nodes in a binary tree that have two children. **Tree** has the following structure.

- The atom **nil** is the empty tree
- **bt(Left, Data, Right)** is a binary tree, where **Left** and **Right** are binary trees

Your solution must use an accumulator. Use **cut** to eliminate multiple counts.

6.

Define a predicate **add_up_list(L, K)** that asserts that Land 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
```

7.

Write a Prolog predicate `countBT(Tree, Count)` to count the number of nodes in a binary tree that have two children. 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. You must document your solution.

8.

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.

9.

Write a Prolog predicate `countOneChild(Tree, Count)` that defines the number of nodes in a binary tree that have exactly one child. 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. You must document your solution.

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 not to prevent extra answers.

11.

A Write a predicate, `count_non_leaf (Tree, Count)`, that is true if `Count` is the number of non-leaf nodes in the Tree. Use cut and/or not to eliminate multiple answers. Use an accumulator. The empty tree is represented by a un-instantiated variable. The predicate var (X) returns true if X is an un-instantiated variable and false otherwise. A tree node is defined by `bt(left-tree, data, right-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.

Define a predicate `memCount(AList,Blist,Count)` that is true if `Alist` occurs `Count` times within `Blist`. You must use an accumulator. Use "not" to make similar cases unique.

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
```

14.

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 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 = 1 ;
no

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

15.

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

16.

B 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.