CS326 – Programming Languages, Concepts and Implementation

# Homework 7

# (Due December 1)

**Submission**: along with the usual paper submission at the beginning of the class, also send an email to cs326@cse.unr.edu containing:

- subject: HW 7
- message body: your name
- attachment: the electronic version of your code (only one file with all predicates); the file must be able to load and be tested in the interpretor

For all problems below, you only need to make sure that the predicates correctly find the first solution, as in the examples provided. You do NOT need to worry about what happens when asking for more solutions (with "`;`").

1. (14 pts) Write the rules for a predicate `reverse(L,L1)`, which succeeds if list `L1` is the list L reversed. The following query shows an example of using this predicate:

```
?- reverse([1,2,3], L1).
L1 = [3,2,1]
```

2. (15 pts) Write the rules for a predicate `take(L,N,L1)`, which succeeds if list `L1` contains the first N elements of list L, in the same order. The following queries show examples of using this predicate:

```
?- take([5,1,2,7], 3, L1).
L1 = [5,1,2]
?- take([5,1,2,7], 10, L1).
L1 = [5,1,2,7]
```

3. (56 pts) Consider the following definition of a binary tree in Prolog, where a tree is either the constant `nil`, or a structure `node` with 3 elements, the second and third elements also being trees:

```
tree(nil).
tree(node(_,Left,Right)) :- tree(Left), tree(Right).
```

(a) (14 pts) Write the rules for a predicate `nleaves(T,N)`, which succeeds if `N` is the number of *leaves* in the tree `T`. The following query shows an example of using this predicate:

```
?- nleaves(node(1, node(2, node(3,nil,nil), node(4,nil,nil)),
node(5,nil,nil)), N).

N = 3
```

(b) (14 pts) Write the rules for a predicate `treeMember(E,T)`, which succeeds if `E` appears as an element in the tree `T`. The following query shows an example of using this predicate:

```
?- treeMember(3, node(1, node(2, node(3,nil,nil),
node(4,nil,nil)), node(5,nil,nil))).

Yes
```

(c) (14 pts) Write the rules for a predicate `preOrder(T,L)`, which succeeds if `L` is a list containing all elements in the tree `T` corresponding to a pre-order traversal of the tree. The following query shows an example of using this predicate:

```
?- preOrder(node(1, node(2, node(3,nil,nil), node(4,nil,nil)),
node(5,nil,nil)), L).

L = [1, 2, 3, 4, 5]
```

(d) (14 pts) The *height* of a tree is defined as the maximum number of nodes on a path from the root to a leaf. Write the rules for a predicate `height(T,N)`, which succeeds if `N` is the height of the tree `T`. The following query shows an example of using this predicate:

```
?- height(node(1, node(2, node(3,nil,nil), node(4,nil,nil)),
node(5,nil,nil)), N).

N = 3
```

You may want to use the predefined arithmetic function `max(X,Y)`.

4. (15 pts) Write the rules for a predicate `insert(X,L,L1)`, which succeeds if list `L1` is identical to the sorted list `L` with `X` inserted at the correct place. Assume that `L` is already sorted. The following query shows an example of using this predicate:

```
?- insert(5, [1,3,4,7], L1).

L1 = [1,3,4,5,7]
```

5. (Extra Credit - 10 pts) Write the rules for a predicate `flatten(A,B)`, which succeeds if `A` is a list (possibly containing sublists), and `B` is a list containing all elements in `A` and its sublists, but all at the same level. The following query shows an example of using this predicate:

```
?- flatten([1, [2, [3, 4]], 5], L).

L = [1, 2, 3, 4, 5]
```