Logic Programming

Prolog Problems: Practical Test

January 10, 2013

1. Write a predicate swap_1_3 that accepts a list and generates from it a similar list with the first and the third element swapped. Shorter lists should be left untouched. Sample runs:

```
?- swap_1_3([],[]).
   true.
?- swap_1_3([1, 2], X).
   X = [1,2].
?- swap_1_3([a,b,c,d], X).
   X = [c, b. a, d].
```

2. Write a predicate for reversing lists, using accumulators.

3. Write a predicate for determining the maximum element from a list of integers.

4. Write a predicate for determining the minimum element from a list of integers.

5. Write a predicate swap_every_2 for swapping every consecutive pair of elements from a list. Sample run:

```
?- swap_every_2([1, 2, 3, 4, 5, 6], X).
X = [2, 1, 4, 3, 6, 5].

?- swap_every_2([1, 2, 3, 4, 5, 6, 7], X).
X = [2, 1, 4, 3, 6, 5, 7].
```

6. Write a predicate that detects flat lists, i.e. lists that do not contain other lists as elements.

7. Write a predicate prefix that gives the list of all prefixes of a list. Sample run:

```
? - prefix([a, b], X).
   X = [[],[a], [b], [a, b]].
```

8. Write a predicate suffix that gives the list of all suffixes of a list. Sample run:

```
? - suffix([a, b], X).
   X = [[], [b], [a,b]].
```

9. Define a predicate sublist that gives sublists of a list. Sample run:

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

10. Write a predicate merge that takes two sorted lists and merges them in a sorted list. Check that the lists are sorted. Sample run:

```
? - merge([1, 4, 6, 7], [2, 4, 5, 6], X).
  X = [1, 2, 4, 4, 5, 6, 6, 7].
```

11. Write a predicate remove_at(X, L1, K, R)for the removal of the Kth element X from a list. Sample run:

```
?- remove_at(X,[a,b,c,d],2,R).
X = b
R = [a,c,d]
```

12. Write a predicate three_times(L1, L2) that succeeds if the list L2 is three times as long as the list L1.

13. Write a predicate flatten (L1, L2) such that L2 is the flattened version of L1. Sample run:

```
?-flatten([[1], [1, [2, 3], [2]], [4, 5, 6]],X).
  X = [1, 1, 2, 3, 2, 4, 5, 6].
```

14. Write a ternary predicate count_occurences(El, List , N) that gives the number N of occurences of El in the list List. Sample run:

```
?-count_occurences(2, [a, b, 2, 3, s, 2, 2], N).
 N = 3.

?-count_occurences(z, [a, b, 2, 3, s, 2, 2], N).
 N = 0.
```

15. Write a ternary predicate delete_nth that deletes every N'th element from a list. Sample runs:

```
?- delete_nth([a,b,c,d,e,f],2,L).
 L = [a, c, e].

?- delete_nth([a,b,c,d,e,f],1,L).

 L = [].

?- delete_nth([a,b,c,d,e,f],0,L).
false.
```

```
?- delete_nth([a,b,c,d,e,f],10,L).
 L = [a, b, c, d, e, f].
```

16. Write a predicate for splitting a list into three lists of equal size (or almost equal, i.e. the difference between the length of any of the splits should not be more than 1) .

17. Write a predicate eliminate_consecutive for the elimination of consecutive duplicates in a list. Sample run:

```
?- eliminate_consecutive([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
 X = [a,b,c,a,d,e].
```

18. Define a binary relation last between lists and their last element.

19. Define the binary relation last_but_one between a list its last but one element. Sample run:

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

20. Write a predicate pack that packs consecutive duplicates in sublists. Sample run:

```
?- pack([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [[a,a,a,a],[b],[c,c],[a,a],[d],[e,e,e,e]].
```

21. Write a predicate for taking the even numbers from a list and writing them into a file.

22. Represent sets as lists. Write a predicate that computes the union of two sets. Make sure that the answer is a set! (No duplicate elements.)

23. Represent sets as lists. Write a predicate that computes the intersection of two sets. Make sure that the answer is a set! (No duplicate elements.)

24. Represent sets as lists. Write a predicate for the (weak) subset relation. Make sure that the answer is a set! (No duplicate elements.)

25. Represent sets as lists. Write a predicate for the strict subset relation. Make sure that the answer is a set! (No duplicate elements.)

26. Represent sets as lists. Write a predicate for the set difference of two sets. Make sure that the answer is a set! (No duplicate elements.)

27. Represent sets as lists. Write a predicate that detects whether two sets are equal.

28. Represent sets as lists. Write a predicate that gives the cartesian (cross) product of two sets.

29. Write a predicate that determines whether two lists are permutations of eachother.

30. Write a predicate that replaces every "A" in a string by "a".

31. Write a predicate that deletes all vowels from a string.

32. Write a predicate insert for the insertion of an element in a sorted list such that the result remains sorted. Sample run:

```
?- insert(3, [1, 2, 4, 5, 7, 8], X).
X = [1, 2, 3, 4, 5, 7, 8].
```

33. Write a predicate that reads a term from a file and succeeds if the term is a list.

34. Write a predicate that partitions a list of integers into 3 lists of integers: in the first put the negative numbers, in the second put the odd positives, in the third put the even positives.

35. Write a predicate that detects prime numbers.

36. Write a predicate that implements the following function on real numbers: $f(x) = x^{20} + 5$.

37. Write a predicate that implements the following function on real numbers: $g(x) = \frac{94-3x}{x^2-9}$.

38. Write a predicate that implements the following function on real numbers: $h(x, y) = 5x^{35}\sqrt{y} + 8x^4y^2$.

39. Write a predicate that determines whether two integers are coprime (they have no common divisor except 1).

40. Write a predicate that determines the sum of the square of the elements in a list of numbers. Use accumulators.