**Exercise 4.1**

How does Prolog respond to the following queries?

1. [a,b,c,d] = [a,[b,c,d]]. No
2. [a,b,c,d] = [a|[b,c,d]]. Yes
3. [a,b,c,d] = [a,b,[c,d]]. No
4. [a,b,c,d] = [a,b|[c,d]]. Yes
5. [a,b,c,d] = [a,b,c,[d]]. No
6. [a,b,c,d] = [a,b,c|[d]]. Yes
7. [a,b,c,d] = [a,b,c,d,[]]. No
8. [a,b,c,d] = [a,b,c,d|[]]. Yes
9. [] = \_. Yes
10. [] = [\_]. No
11. [] = [\_|[]]. No

**Exercise 4.2**

Suppose we are given a knowledge base with the following facts:

tran(eins,one).

tran(zwei,two).

tran(drei,three).

tran(vier,four).

tran(fuenf,five).

tran(sechs,six).

tran(sieben,seven).

tran(acht,eight).

tran(neun,nine).

Write a predicate listtran(G,E) which translates a list of German number words to the corresponding list of English number words. For example:

listtran([eins,neun,zwei],X).

should give:

X = [one,nine,two].

Your program should also work in the other direction. For example, if you give it the query

listtran(X,[one,seven,six,two]).

it should return:

X = [eins,sieben,sechs,zwei].

Hint: to answer this question, first ask yourself ‘How do I translate the *empty* list of number words?’. That’s the base case. For non-empty lists, first translate the head of the list, then use recursion to translate the tail.

listtran([],[]).

listtran([G|Gs],[E|Es]) :- tran(G,E), listtran(Gs,Es).

**Exercise 4.3**

Write a predicate twice(In,Out) whose left argument is a list, and whose right argument is a list consisting of every element in the left list written twice. For example, the query

twice([a,4,buggle],X).

should return

X = [a,a,4,4,buggle,buggle]).

And the query

twice([1,2,1,1],X).

should return

X = [1,1,2,2,1,1,1,1].

Hint: to answer this question, first ask yourself ‘What should happen when the first argument is the *empty* list?’. That’s the base case. For non-empty lists, think about what you should do with the head, and use recursion to handle the tail.

twice([],[]).

twice([X|Xs],[X,X|Ys]) :- twice(Xs,Ys).

**Exercise 4.4**

Draw the search trees for the following three queries:

?- member(a,[c,b,a,y]).

Call: (7) member(a, [c, b, a, y])

Call: (8) member(a, [b, a, y])

Call: (9) member(a, [a, y])

Exit: (9) member(a, [a, y])

Exit: ...

?- member(x,[a,b,c]).

Call: (7) member(x, [a, b, c])

Call: (8) member(x, [b, c])

Call: (9) member(x, [c])

Call: (10) member(x, [])

Fail: (10) member(x, [])

Fail: ...

?- member(X,[a,b,c]).

Call: (7) member(\_G321, [a, b, c])

Exit: (7) member(a, [a, b, c])

X = a ;

Redo: (7) member(\_G321, [a, b, c])

Call: (8) member(\_G321, [b, c])

Exit: (8) member(b, [b, c])

Exit: …

X = b ;

Redo: (8) member(\_G321, [b, c])

Call: (9) member(\_G321, [c])

Exit: (9) member(c, [c])

Exit: …

X = c ;

Redo: (9) member(\_G321, [c])

Call: (10) member(\_G321, [])

Fail: (10) member(\_G321, [])

No