1. Create a new Prolog programme file in your Prolog programmes directory and call it `lab3.pl`. Add to it the definition of `tmember` given in Lecture 9 (the lecture on lists in Prolog). Use `tmember` to find whether:

   (a) 3 is a member of the list [1,2,3,4,5]
   (b) 7 is a member of the list [1,2,3,4,5]
   (c) a is a member of the list [l,a,b,o,r,a,t,o,r,y]
   (d) z is a member of the list [l,a,b,o,r,a,t,o,r,y]

2. Try tracing each of the above examples. (See last week’s exercise for details on how to perform a trace).

3. From the notes for Lecture 9, add the definition of `tappend` to your programme and reload. Note what results you get for the following queries:

   (a) `tappend([3,4,5],[1,2],X).`
   (b) `tappend([1,2],[3,4,5],X).`
   (c) `tappend([3,4,5],[],X).`
   (d) `tappend([],[3,4,5],X).`

   Trace your solutions as before.

4. Write a rule `sumList(X,Y)` which, when a list of numbers X is input, Y will contain the sum of that list. Use `sumList(X,Y)` in queries to verify that it works as expected.

5. Write a rule `deleteOne(X,Y,Z)`, such that Z is the result from deleting the first occurrence of Y in the list X. If X does not contain Y then just return X. Use `deleteOne(X,Y,Z)` in queries to verify that it works as expected.

6. Write a rule `deleteAll(X,Y,Z)`, such that Z is the result from deleting all occurrences of Y in the list X. Use `deleteAll(X,Y,Z)` in queries to verify that it works as expected.

7. Write a rule `reverseList(X,Y)`, such that given list X, Y is the result of reversing X. Use `reverseList(X,Y)` in queries to verify that it works as expected.

8. Write a rule `oddEven(X,Y)`, such that Y is set to “odd” if the number of elements in the list X is odd and Y is set to “even” if the number of elements in the list X is even. Hint: you need two sets of rules (oddEven and oddEven1) each calling the other alternately. Use `oddEven(X,Y)` in queries to verify that it works as expected.