Functional Programming
Exercise set 3

Tony Lopar s1013792
Carlo Jessurun s1013793
Marnix Dessing s1014097

December 7, 2017

Exercise 1

This is implemented in *WordList.lhs*.

Exercise 2

This exercise can be found in the attached file *Pattern.lhs*.

Exercise 3

The implementation of the run function may be found in the attached file *Run.lhs*.

The partitioned list already contains partially sorted lists. So, when we want to sort the list, we already know that some parts are already sorted which decreases computing time.

Exercise 4

Exercise 5

1. The implementation can be found in the attached file *Quicktest.lhs*. In order to check whether the next element in non-decreasing we should check that the next value is higher of equal to the previous.

2. The function is defined as permutations. The number of factorations is equal to the factorial of n, so permutations n will result a list of length n!.

3. A possible testing procedure could be to check for each element of the list whether it's in a non-decreasing order.

4. We can use the fact that $\sqrt{n} \cdot \sqrt{n} = n$ to check whether the right square root has been computed.

5. If the first list contains n and the second list m elements there will be n x m combinations possible, since every value from n can be combined with all m. So if we have bools = [True, False] and chars = "Tony" we will have 8 pairs in a list as result when we execute $bools \otimes chars$ since bools contains 2 and chars contains 4 elements. The output will be: [(True,'T'),(True,'o'),(True,'n'),(True,'y'),(False,'T'),(False,'o'),(False,'n'),(False,'y')]

Exercise 6