Exercises- Week 4

Building lists

Part I

1. Given a list of natural numbers, display all even elements.

2. Given a list of strings, display all elements that are less than 3 characters long.

3. A list of several data types (integers, reals, strings, booleans) is given. Sort the list ascending, then descending. (explanation: when running it will give an error because lists with elements of different types cannot be sorted. number<string of characters cannot be compared. Only lists containing the same type of elements can be sorted)

4. Sort a list containing integers according to the last digit of the number (in ascending order of the last digit of the number).

5. Multiply all elements of a list of real numbers. (hint: the reduce() function can be used)

6. Create a new list containing all the elements of an initial list to the power of 3. The map() function will be used.

7. Create a new list containing only the prime number elements from an initial list. The filter() function will be used.

Part II

1. a) Implement functions that build the list of digits of a number that satisfy a given condition (odd, even, less than 7 digits, etc. of your choice), in normal and reverse order

b) Implement a function that builds the list of digits of a number that satisfy a condition given as a parameter in the form of a function of type int -> bool.

c) Conversely, given a list of digits, construct the number consisting only of the digits that meet a condition (given as a function parameter with the type int -> bool). Solve the problem directly, recursively, and then by using filter (for digit selection) along with reduce(for building the number).

2. Implement the function fromto that generates the list of integers in a given range that are divisible by a given value. (hint: find the largest divisible number in the range, and continue step by step)

3. a) Implement the function nth, which returns the nth element in a list.

b) Implement a function firstn, which returns a list of the first n elements from a given list.

Traversing list elements

5. a) Implement a function called filter, with the same behavior as the default filter function, using reduce().

b) Implement the function exists which determines (returns true/false) whether there is an element in the list that satisfies a condition (a function taking 1 parameter and returning a bool given as a parameter).

6. a) Using reduce, implement a function called count, which takes a function f and a list as a parameter and returns the number of elements for which the function f is true

b) Similarly, implement a function sum which calculates the sum of all elements (assumed integers) for which the function f it's true.

7. Implement the functions split and combines which turn a list of pairs into a pair of lists, and vice versa.

Example: split([(1,2), (3,4), 5,6)]) -> ([1,3,5], [2,4,6])

combine([1,3,5], [2,4,6]) -> [(1,2), (3,4), 5,6)]

8. Implement the function PARTITION which takes as parameters a condition as a function and a list and returns a pair of lists, with the elements that satisfy and do not satisfy the condition respectively.
Example: partition (lambda x : x >= 5) [4,6,7,5,4,8,9] -> ([6, 7, 5, 8, 9], [4, 4])

9. Write a function that takes a list of digits and returns the value of the number with those digits.

10. Write a function that removes consecutive duplicates: takes a list as parameter and constructs a list in which all sequences of equal consecutive elements have been replaced by a single element.

11. Write a function that compares two lists according to the following ordering relation: a shorter list is "smaller" than a longer one; if the lengths are equal, the ordering is determined by the first pair of different elements. Avoid unnecessary or repeated scrolling through lists. The function will return a negative, 0, or positive integer depending on the ordering of the two argument lists.

Merge sort

12. Write a function that receives two lists as parameters, each sorted ascending, and returns a list containing the elements of both lists in ascending order. Compare the first elements of both lists to decide which will be first in the result, and then continue with the remaining lists.

13. Write a function that splits a list into two lists whose length differs by at most 1 element, by alternately putting one element in each of the lists. (The function will return a pair of lists).

14. Sort a list using the merge sort algorithm. Split the list into two halves, and recursively sort each of them, and then merge the two sorted lists. Use the functions from the previous two problems.