Concurrent and Parallel Programming in Java 8

part I – FP systems

1. Define the function **len**, which returns the length of the list passed as argument, without using the predefined function **length**.

   len:<1,2,3> **len:<>**
   3 **0**

2. Define the function **member**, which returns a Boolean value indicating if the first element belongs to the list passed in the second position of the argument of length two.

   member:<2, <1,2,3>> **member:<6, <>>** **member:<2, <2>>** **member<4, <1,2,3>>**
   T **F** **T** **F**

3. Define the function **count0**, which returns the number of zeroes in a list.

   count0:<1,2,3> **count0:<>** **count0:<3,4,0>** **count0:<0,1,2,0>**
   0 **0** **1** **2**

4. Define the function **concat**, which receives a list of two sub-lists and returns its concatenation.

   concat:<<1,2,3>,<4,5>> **concat:<<1,2>, <>>** **concat:<<>,<>>**
   <1,2,3,4,5> **<1,2>** **<>**

5. Define the function **avg**, which returns the average of the values of a list of integers.

   avg:<3,6,9> **avg:<>** **avg:<4>** **avg:<0,2,2,0>**
   6 **0** **4** **1**

6. Define the function **prod**, which returns the product of the values of a list of integers.

   prod:<3,1,2> **prod:<>** **prod:<4>** **prod:<0,2,2,0>**
   6 **1** **4** **0**

7. Define the function **prod2**, which returns the product of a list of two positive integer values, without using the operator ***x*** (product).

   prod2:<3,4> **prod2:<3,0>** **prod2:<2,1>**
   12 **0** **2**

8. Define the function **rev**, which returns the reverse of a list, without using the function **reverse**.

   rev:<1,2,3,4,5> **rev:<<1,2>, <3,4>>** **rev:<>**
   <5,4,3,2,1> **<<3,4>,<1,2>>** **<>**

9. Define the function **allRev**, which returns a list containing all reversed sub-lists of the original two-levels list.

allRev:<<1,2,3>,<4,5>> **<<3,2,1>,<5,4>>**	**allRev:<<1,2>, <>,<4,5,6>>** **<<2,1>,<>,<6,5,4>>**	**allRev:<<>,<>>** **<>**

10. Define the function **flatten**, which returns the concatenation of all the values of the original two-levels list.

flatten:<<1,2,3>,<4,5>> **<1,2,3,4,5>**	**flatten:<<1,2>, <>,<4,5,6>>** **<1,2,4,5,6>**	**flatten:<<>,<>,<3,4>,<> >** **<3,4>**

11. Define the function **allFlatten**, which returns the concatenation of all the values of the original argument.

allFlatten:<<1,2,3>,<4,5>> **<1,2,3,4,5>**	**allFlatten:<<1,2>, <>,<6,<7>>>** **<1,2 ,6,7>**	**allFlatten:<<>,<>,<<<>,<<>>>>** **<>**

12. Define the function **diag**, which returns the diagonal of the square matrix passed as argument.

diag:<<1,2,3>,<4,5,6>,<7,8,9>> **<1,5,9>**	**diag:<<5,6>, <8,9>>** **<5,9>**	**diag:<>** **<>**

13. Define the function **posAndNeg**, which returns a list with two integers, being the first the summation of all positive values and the second the summation of all negative values, of the original list of integers passed as argument.

posAndNeg:<1,5,-4,3,-2> **<9,-6>**	**posAndNeg:<1,3,6,1>** **<11,0>**	**posAndNeg:<>** **<0,0>**

14. Define the function **posAndNegList**, which returns a list with two sub-lists, being the first the list of all positive values and the second the list of all negative values, obtained from the original list of integers passed as argument.

posAndNegList:<1,5,-4,3,-2> **<<1,5,3>,<-4,-2>>**	**posAndNegList:<1,3,6,1>** **<<1,3,6,1>,<>>**	**posAndNegList:<>** **<<>,<>>**

15. Define the function **quickSort**, which returns the same list of integers passed as argument, but sorted using the quick sort algorithm.

quickSort:<6,1,3,2,9> **<1,2,3,6,9>**	**quickSort:<>** **<>**	**quickSort:<1,1>** **<1,1>**