Scala: Exercises

1) Define variables and immutable variables (i.e., values) using the Scala shell (REPL)
   a) Open the shell by executing the command line command scala
   b) Define a new variable of type **String**. The name of the variable is **message** and its initial content is "**Hi**"
   c) Update the content of **message** with the new value "**Hello**"
   d) Define a new variable of type **String**. The name of the variable is **message2**. Do not initialize the value of message2. What happens?
   e) Define a new immutable variable (i.e., a value) of type **String**. The name of the immutable variable/value is **messageConst**. Set the content of **messageConst** to "**Welcome**"
   f) Update the content of **messageConst** with the new value "**Welcome!!!**". What happens?
   g) Define two variables of type **Integer**. The names of the variables are **x** and **y**. Set **x**=10 and **y**=3. Compute the formula **x/y** and store the result in a new variable **z**. Do not specify the data type of **z** during its definition. What is the data type assigned to **z** by Scala?
   h) Store the result of **x/y** in a new variable **w** of type **Float**. Convert **x** and **y** to **Float** before executing the operation x/y by using **toFloat**

2) Lists, Map
   a) Define a list of integers containing the values 1, 2, 3 and assign it to an immutable variable called **numbers**.
   b) Print on the console the value of the second element of **numbers**.
   c) Assign the value 45 to the second element of **numbers**. What happens?
   d) Define a new variable of type list, called **secondList**, and assign to it the tail of **numbers** (invoke **numbers.tail**)
   e) Add the element 33 at the beginning (head) of the list **numbers** and store the new list in **extendedList**.
   f) Define a Map variable. The name of the variable is **cityProvince**. The data type of the keys is **String**. Also the data type of the values is **String**. Keys = name of Italian cities. Values=Italian provinces. Insert in **cityProvince** the pairs **Torino -> Piemonte** and **Milano -> Lombardia**.
   g) Print on the console the province of Torino (use **cityProvince** to retrieve the province of Torino)
   h) Print on the console the province of Palermo (use **cityProvince** to retrieve the province of Torino). What happens?

3) Expressions and loops
   a) Given a variable **amount** of type Double, write an expression to return the String "greater" if **amount** is greater than zero, "less than or equal to" if **amount** is equal to or less than zero. Store the returned String into the variable **message**. Implement it by using if-else and then by using match expressions.
   b) Define the immutable variables **numbers**=List(4,5,-1). Use the for loop to print on the console the content of **numbers**.

c) Iterate over the content of **numbers** and return (by using the yield option) a list containing a Boolean value for each element of numbers. For each element **e** of **numbers** the Boolean value **true** is inserted in the new list if **e** is greater than 0. Otherwise **false** is inserted in the new list.

4) Functions
   a) Write a function that calculates the first value raised to the exponent of the second value. Both parameters of the function are integers. Try writing this first using math.pow, then with your own calculation. Did you implement it with variables? Is there a solution available that only uses immutable variables/values?
   b) Write a function literal that takes two integers and returns the higher number. Assign it to a "function" variable and invoke it.
   c) Write a function called "conditional" that takes a value **x** of type Int and two functions, **p** and **f**, and returns a value of the same type as **x**. The **p** function is a predicate, taking am integer value and returning a Boolean. The **f** function also takes an integer value and returns a new value of the same type (i.e., an Int). Your "conditional" function should only invoke the function f(**x**) and returns f(**x**) if p(**x**) is true, and otherwise return **x**. Invoke the function "conditional" by specifying the following functions for the arguments **p** and **f**:
      i) **p**: a function literal that returns true if **x** is greater than 15. false otherwise.
      ii) **f**: a function literal that returns **x*2**
      Perform the same invocation by using the placeholder syntax

5) foreach, map, reduce, filter
   a) Define the immutable variables **numbers**=List(4,5,-1). Use the for loop to print on the console the content of **numbers**. Implement an alternative solution to this problem by using foreach and compare it with the solution based on the for loop.
   b) Iterate over the content of **numbers** and return a list containing a Boolean value for each element of numbers. For each element **e** of **numbers** the Boolean value **true** is inserted in the new list if **e** is greater than 0. Otherwise **false** is inserted in the new list. Implement it by using map(). Compare this solution with the one based on the for loop + yield option.
   c) Apply the function **e*2** on all elements **e** of **numbers** and compute the sum of the returned values. Store the sum in the variable **sumValues**. Solve the problem by using three different approaches:
      i) The for loop
      ii) A recursive function
      iii) map() and/or reduce()
   d) Create a list of the first 10 Int numbers (from 1 to 10) and then split it in two sublists: odd and even lists. Use the following approaches to split the list:
      i) for-loop + if
      ii) filter operation
      iii) partition operation
   e) Define the immutable variables **numbersMapReduce**=List(4,5,-1). Write a program that sum the values in **numbersMapReduce** greater than 2. Implement a solution based on map, reduce, and filter.

f) Define the immutable variables **numbersMR**=List(4,5,-1). Write a program considers only the values greater than 2 occurring in **numbersMR**. Then the program computes the square of the selected values. Implement a solution based on map, reduce, and filter.
g) Write a code that takes a list of strings and returns the longest string in the list. Avoid using mutable variables. Use reduce().

6) Object-oriented programming
   a) Create a new Scala project in Eclipse
   b) Implement a command line application that prints on the console the content of its arguments
   c) Define a class representing courses. The name of the class is Course. Each instance of Course is characterized by the title of the course, number of hours, maximum number of students.
   d) Write a primary constructor that allows specifying all the fields of Course.
   e) Write an auxiliary appropriate constructor such that the maximum number of students is not a parameter of the constructor and it is automatically set to 10.
   f) Define two variables of type Course and initialize them. Initialize the first variable by creating an instance of Course for which all the characteristics are specified and initialize the second variable with another instance of Course for which the maximum number of students is not specified.
   g) Print of the console the titles of the two courses.
   h) Override the toString method of Course. The toString method must return the string “Course: title of the course, Hours: num. of hourse, Students: num .of students ”.
   i) Store the two courses in a variable of type List[Course] called myCourses
   j) Iterate over the myCourses and count the number of courses with “maximum number of students” greater than 15. Compute also the sum of the hours of the courses with “maximum number of students” greater than 15. Implement a solution based on filter, map, fold. What happens if you use reduce() instead of fold() and there is no courses satisfying the constraint “maximum number of students” greater than 15?