**PLF - Laborator 1 - R1**

Details ... from <http://www.cs.ubbcluj.ro/~hfpop/teaching/pfl/requirements.html>

and <http://www.cs.ubbcluj.ro/~hfpop/teaching/pfl/contents.html>

**7. Lab activity**

**Lab papers and grading**

**(1)** The lab problem will be presented to the lab instructor through **shared screen** when formally delivered. At that moment an extra request to change the code will be solved on the spot.

**(2) Each lab paper is evaluated with a grade from 1 to 10 in the following way:**

* 2 points: formal descriptions and explanations of recursive models
  + PROLOG - recursive models and flow models of all predicates used, meaning of all predicate parameters;
  + LISP - mathematical models (recursive formulas describing functions), meaning of all function parameters;
* 0.50 points: source code in Prolog/Lisp for all predicates / functions
* 0.50 points: testing examples covering as many testing cases as possible, for the essential predicates / functions
* 1 points: execution verification of the written program
* 6 points: explanations of the written algorithm, recursive models, source code
  + extra requirement to change the code, solved on the spot, with shared screen

**(3) A copied lab paper** means cheating and will be graded with 0 (zero).

**(4) If a lab paper is submitted with another lab subgroup** inside the same two weeks frame, the lab grade is **multiplied by 0.8**; if the submission is delayed one lab, the lab grade is **multiplied by 0.6**; if the delay is larger than one lab, the final grade is **0 (zero)**.

**8. Lab attendance and delivery rules**

**(1) Attendance at lab activities is compulsory** in proportion of 90% (minimum 6 labs out of 7).

**(2) The lab attendance may be redone** for at most **one** lab, during the two weeks allocated to that lab, with the explicit agreement of the lab instructor, but the lab doc is graded in the usual way, considering the delay penalty. In case of illness, the absence is justified to the lab instructor based on medical evidence only. Medical documentation will be provided with **a maximum delay** of one week after the missded event.

**(3) The students without minimum 6 attendances at laboratory activities** CANNOT participate to the written paper (during weeks 13-14 and during the resit session) and CANNOT pass the exam.

**(4) At most two lab themes** may be delivered during one lab class. As well, the Prolog labs may be delivered until no later than ween 7/8 (or 9/10).

**(5) The lab grade will be determined as the average** of the grades of all the lab works.

If a lab doc is not delivered, its grade is 0.

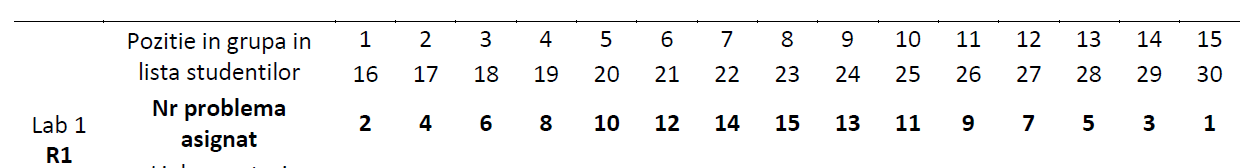
**Essentially:**

At each lab, you will receive one problem to solve and present.

For our first lab, the list of problems is here:

<http://www.cs.ubbcluj.ro/~hfpop/teaching/pfl/lab/r1.pdf>

The delivery for Lab1-R1 is week 3/4.



The mathematical models can be written on the paper or in an online editor and the code for this lab (R1) can be written in whatever programming language do you want.

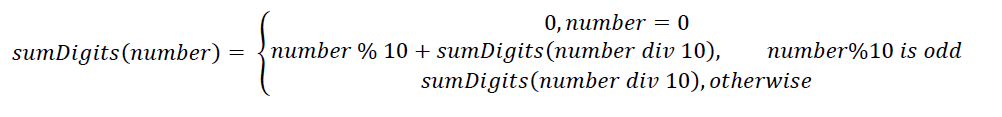
**R1: Recursive programming**

The aim of this lab is for you to get familiar with recursive thinking and recursive problem solving that will be required later both in Prolog (logic programming) and Lisp (functional programming). A recursive solution to a problem can be described using a “recursive mathematical model” or “recursive model”. This is a mathematical description of your solution.

**Example**: Determine recursively the sum of the odd digits of a number: sumDigits(123) = 4, sumDigits(41708) = 8, etc.

**Solution:** We can determine the last digit of a number using the modulo operator and by dividing the number by 10 we can get rid of the last digit. If the last digit is even, we don’t have to consider it, in this case we simply determine recursively the sum of digits for the number without the last digit. If the last digit is odd, we will add it to the sum of digits computed recursively for the rest of the number. We will stop when the number becomes 0, in this case the result is 0.

This solution process can be written with a recursive model in the following way:



All problems from this lab (and most problems from the other labs, as well) require work with lists (or sets, but a set is simply a list with unique elements). In the recursive model, lists are represented by enumerating the elements: l1l2l3...ln.

When working with lists, we only have a few available operations, and there are many things we cannot do:

- We don’t have access to the length of a list. If we really need to know the length of the list, we have to write a recursive function to compute it.

- However, we can check whether the list contains a given number of **constant** elements. We can check whether:

* the list is empty: n = 0
* the list has only one element: n = 1
* the list has only two elements: n = 2
* etc..

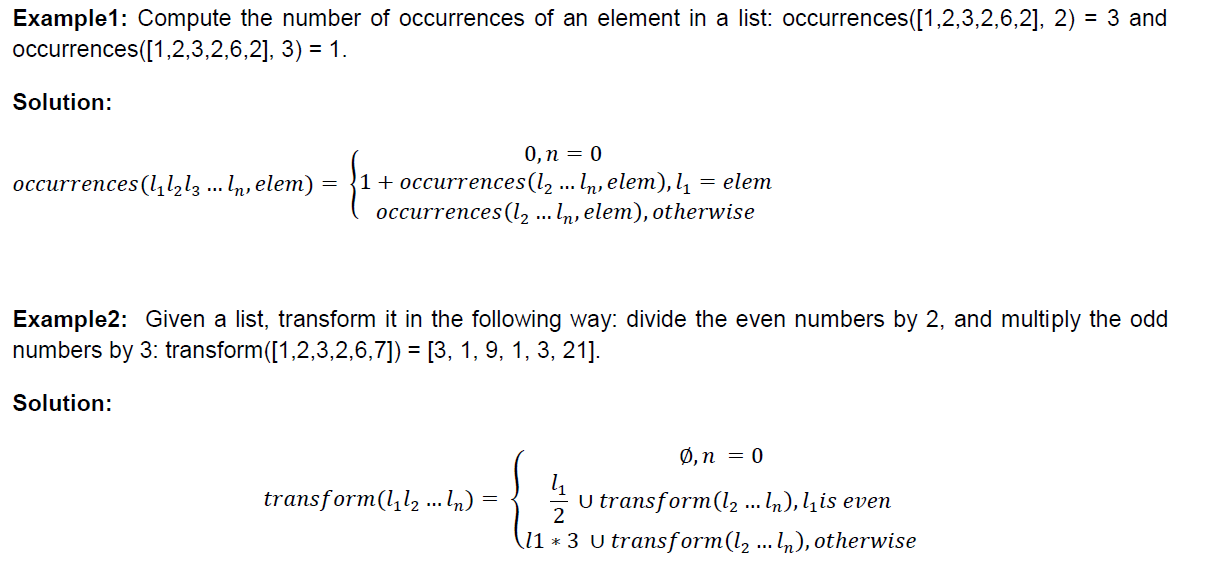
- We **cannot** check whether the length of the list is equal, less than or greater than a value that is not constant.

- We can only access elements from the beginning of the list, and only a **constant** number of elements. When we access an element from the beginning of the list, we also have access to the rest of the list:

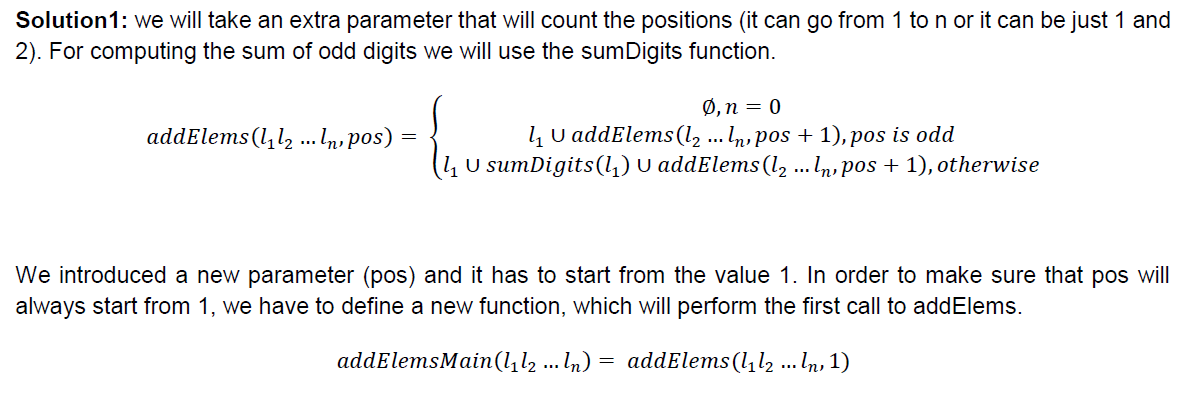
* l1 is the first element and l2l3…ln is the rest of the list
* l2 is the second element and l3…ln is the rest of the list
* etc…

- We **cannot** access the last element of a list and we cannot access elements from the middle of the list (elements whose position is not a constant, ex. lk, where k is a parameter of the function)

- When the solution of a problem is a list, we will create a new list (we cannot modify the list that we have as parameter) and add to this list the elements that we need using the reunion operator: U. We can only add element(s) to a list and only to the beginning of the list. We cannot concatenate two lists and cannot insert an element to an arbitrary position (unless we write specific functions that do this).

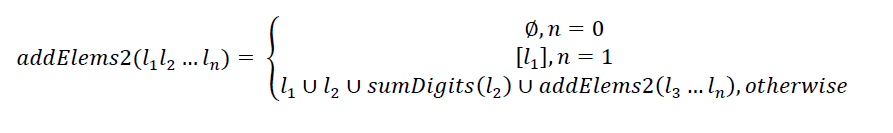


**Example3:** Given a list, add after every element from an even position the sum of its odd digits. The first element of the list is on position 1: addElems([1,2,13,65,297,543,63]) = (1, 2, **0**, 13, 65, **5**, 297, 543, **8**, 63]

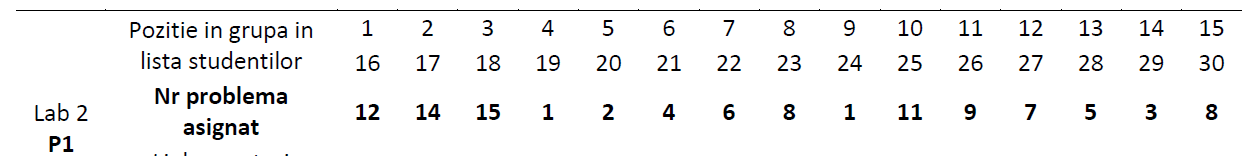


Solution2:

If the list has only one element that element is at an odd position (position 1)



Starting with the problems from Laborator 2 - P1, the code will be implemented in Prolog



The list of problems for Lab 2-P1 is here:

<http://www.cs.ubbcluj.ro/~hfpop/teaching/pfl/lab/p1.pdf>

The delivery for Lab2-P1 is week 5/6.

If you want to check materials from other year: <http://www.cs.ubbcluj.ro/~adrianac/PLF/>