Principi di Linguaggi di Programmazione
Programming Paradigms

prof. M. Bellia
Regular Exam VI - sept. 10^{th} 2014
(Available Time: 2 hours. Mandatory: In each exercise get at least, one half of the assigned points)

Exercise 1 (pts 10)
Let filter: 'a list -> ('a -> boolean) -> 'a list be the well known operator, on generic lists, that given a predicate as its second argument, returns the list without the elements for which the predicate does not compute true. For this operator, provide in Caml:

1. A recursive definition;
2. A tail-recursive definition;
3. A (functional) iterative definition.

Exercise 2 (pts 10)
Let Tree be an ADT for immutable, finite, generic, trees with vertices labelled with values of a generic type A. The ADT provides the following public operations:

- *cons* is a 2-argument constructor that applies to a value v_A of type A and to a finite list of trees. It returns if possible, a generic tree rooted on a vertex that is labelled with v_A and has, as sons, the trees (in the same order in which they occur) in the list. It raises an exception otherwise.
- *empty* is 0-argument constructor that returns the empty tree of generic type.
- *root* is 1-argument operator that applies to a tree and returns the label of the tree root, if any. It raises an exception otherwise.
- *sons* is 1-argument operator that applies to a tree and returns the son list of the tree root, if any. It raises an exception otherwise.
- *treeAt* is 2-argument operator that applies to a tree and to a finite list of integers. It uses the list as a path in the tree: The first integer n_1 selects the n_1-th son, if any, of the tree and the next ones form a path in the selected son. The empty path is the empty list. The operator returns the tree selected as son, if any, at the end of the path. It raises an exception otherwise.

1. Provide one Caml definition of the API;
2. Provide one Caml definition of an ADT implementing the API defined in (1).
3. Using the API, provide a definition of the function *pathLabel* that applies to a tree of a generic A, and to an integer list. It returns a list of A's containing the labels of the roots of the sons, if any, that are visited by the path specified in the list.

Exercise 3 (pts 10)

1. Show a Java abstract class *TreeJ* for the API of Exercise 2 except that now the trees are mutable values.
2. Let *TreeJADT* be an extension of TreeJ providing an implementation for TreeJ. Define a class *TreeJADTE* that extends *TreeJADT* with an additional public operation *anEq* that applies to two trees and returns true if and only if the two trees have same shape and same labels on the corresponding vertices.
3. Define a class *TreeJADTG* that extends (and specializes) *TreeJADT* to trees having distinct vertices labeled with distinct labels.