Principi di Linguaggi di Programmazione
Programming Paradigms

prof. M. Bellia
End Term Exam - May 30 2014
(Available Time: 2 hours)

Exercise 1 (pts 5)
Use Data Extensions through Functional Abstractions for defining, in Ocaml, data that behave as lists. The new data have the following operations:

empty() that returns the empty list;
cons(u,l) that returns a list with head element u and tail list l;
hd(l) that returns the head element of l, if any;
tl(l) that returns the tail list, if any.

Remind that You can't use structured types of any sort.

Exercise 2 (pts 10)
1) Provide a definition in Ocaml of the function append of two lists using:
   a) recursive technique;
   b) *tail-recursive technique;
   c) iterative technique
2) Which of the above techniques is compatible with lists defined in the previous exercise and why?

Exercise 3 (pts 11)
Let Rel be an abstract data type for mutable, binary relations, of generic type, in Java. Rel has the following public operations:

create() that return an empty relation;
add(x,y) that adds a new pair of elements
reset(x,y,r) that replaces y with r in the pair(x,y)
get2(x) that returns the LinkedList of all the y occurring as 2nd element of pairs having x as the first element.

(a) Show the abstract class defining the signature of Rel;
(b) Show a class RelA defining an ADT for Rel
(c) Show a class RelB that extends RelA with the new public operations:
IsIn(x,y) that return true iff the pair (x,y) belongs to the relation
Size that returns the number of the different pairs in the relation

Exercise 4 (pts 4)
Show a class RelX that extends the class RelA, in the previous exercise, in a type for only symmetric relations. The class has an additional public operation:

get1(y) that returns the LinkedList of all the x occurring as 1st element of pairs having y as the second element.