Indicazioni sula soluzione dell'appello I di PLP:Paradigmi del 16/1/2012
esercizio 1 -- semplice
esercizio 2
(a) module type TREEk =
 sig type ('a,'b) treek
  val treekN: int -> ('a,'b) treek
  val treekE: int -> 'b -> ('a,'b) treek
  val treekR: int -> 'a -> (('a,'b) treek) list -> ('a,'b) treek
  val setSubTree: ('a,'b) treek -> int list -> ('a,'b) treek -> ('a,'b) treek
end;;
(b) module Treek1 =
 (struct
  type ('a,'b) treek = N of int | E of int * 'b | R of int * 'a * (('a,'b) treek)list;;
  let treekN n = (N n);;
  let treekE n b = E (n, b);;
  let rec treekR n a l = if check(n, l) then R(n, a, l) else raise (Failure "treekR") and
          check(n, l) = let f x = match x with
                (N k) when k=n -> true
                |E(k,_) when k=n -> true
                |R(k,_,_) when k=n -> true
                |otherwise ->false
                        in (List.length(l)<=n) && List.fold_right(&&)(List.map f l)true;;
  let rec setSubTree u c v = match (u,c) with
        (R(k,a,l),p::[]) when (p>0 && p<=k && null(getSon u p)) -> R(k,a,replace(l,p,v,k))
        |(R(k,a,l),p::pr) when (p>0 && p<=k) -> let t1=(getSon u p)
                        in let t2=(setSubTree t1 pr v)in R(k,a,replace(l,p,t2,k))
        |otherwise -> raise (Failure "setSubTree")
  ... (* completare con le operazioni ausiliarie null, replace, getSon di ovvio significato *)

end:TREEk);;
(c) -- banale: si applica le definizioni dei costruttori richiamati nell'espressione data.
esercizio 3
(a)Vector<int>
(b)public abstract class TREEk <A,B extends A> {protected int k;}
   public class TREEkN <A,B extends A> extends TREEk <A,B> {code1}
   public class TREEkE <A,B extends A> extends TREEk <A,B> {private B label;code2}
   public class TREEkR <A,B extends A> extends TREEk <A,B> {private A label;
                                                            private Vector<TREEk <A,B>> sons;code3}
(c)code1: public TREEkN(int k){this.k = k;}
   code2: public TREEkE(int k, B label){this.k = k; this.label = label;}
   code3: public TREEkN(int k, A label, Vector<TREEk <A,B>> sons)throws FailureException{
                if (!check(sons,k)) throw new FailureException("TREEkR");
                this.k = k;this.label = label;this.sons = sons;}
          private static <A,B extends A> boolean check(Vector<TREEk <A,B>> sons, int k){
                if ((sons==Null)||(sons.size==0)) return true;
                for(int i=0;i<sons.size();i++){if(sons.get(i).getDegree()!=k) return false;}
                return true;}
 (d) -- semplice: si completa ognuna delle tre classi concrete con la definizione di setSubTree
     adeguata per quella classe.