Linguaggi di Programmazione con Laboratorio
Corso di Laurea in Matematica
Appello Ordinario del 18 Luglio 2019

Allegati

Esercizio 1.

```
module type TREE =
 sig type 'a tree
   exception NoNode of string
   val empty: unit -> 'a tree
   val root: 'a -> 'a tree
   val isEmpty: 'a tree -> bool
   val addSons: 'a tree -> 'a tree list -> 'a tree
   val getRoot: 'a tree -> 'a
   val getSons: 'a tree -> 'a tree list
   val getLabels: 'a tree -> 'a list

end;;



module treeS =
(struct
   ...
(*
   AF(c) = ...
   I(c) = ...
 *)
(* definizioni costruttori ed altre operazioni: omesse *)
   let addSons ...
 end:TREE);;
```

```
/*
empty, root: Costruttori.
isEmpty: Predicato per albero vuoto.
addSons: Aggiunge a destra, nella lista dei figli della radice dell'al-
     bero dato, la lista di alberi fonita come secondo argomento. Se la
     lista non può essere inserita interamente, allota restituisce l'al-
     bero dato.
getRoot, getSons: Accesso ai componenti.
getLabels: Calcola la lista delle etichette presenti nell'albero.
*/
```

Esercizio 2

```
public interface APITree<A> extends Cloneable{
     public boolean isEmpty();
     public void addSons(Vector<APITree<A>> list);
     public A getRoot();
     public Vector<APITree<A>> getSons();
     public Vector<A> getLabels();
     public APITree<A> clone();
     public boolean equals(Object o);
     public String toString();

}



public class TreeS<A> implements APITree<A>{
     /* definizione del corpo della classe omessa */
}



public class TreeSE ...{
     private int limit;
     /*
      AF(c) =...
      I(c) = ...
     */

     /* costruttori */
     public TreeSE(){
           ...
     }
     public TreeSE(A u){
           ...
     }
     public TreeSE(int k, A u){
           ...
     }

     /* definizione altri metodi omessa */
     public void addSons(Vector<APITree<A>> list){
           ...
     }
}
```