Linguaggi di Programmazione con Laboratorio
Corso di Laurea in Matematica
Appello Ordinario del 02 Settembre 2019

Allegati

Esercizio 1.

```
module type STACK =
 sig type 'a stack
   exception Error
   val mk: int -> 'a stack
   val isEmpty: 'a stack -> bool
   val size: 'a stack -> int
   val push: 'a stack -> 'a -> 'a stack
   val top: 'a stack -> 'a
   val pop: 'a stack -> 'a stack
   val remove: 'a stack -> int -> 'a stack
end;;
```

```
module StackB =
(struct
   type 'a sk = E | SK of 'a stack * 'a
   type 'a stack = M of int * ('a sk)
(*
   AF(c) = ...
   I(c) = ...
 *)
(* definizioni costruttori ed altre operazioni: omesse *)
   let push ...
   let remove ...
 end:STACK);;
```

```
/* Nota sul comportamento atteso per le operazioni di STACK e StackB
mk: Costruttore per stack limitati.
isEmpty: Predicato per vuoto.
size: Numero di elementi nello stack
push, top, pop: Le operazioni fondamentali degli stack.
remove: indicato con n il secondo argomento, rimuove l'n-esimo elemento
     a partire dal top dello stack. Se un tale elemento non esiste, sol-
     leva eccezione.
     Esempio.
     Sia AF(s) = [3, 12, 1, 17, 9]. AF(remove s 3) = [3, 12, 17, 9].
*/
```

Esercizio 2

```
public interface MuCloneable extends Cloneable{
    public Object clone();
}

public interface API<A extends MuCloneable> extends MuCloneable{
    public boolean isEmpty();
    public A get() throws EmptyQueueException;
    public void add(A x);
    public void addAll(Vector<A> els);
    public void remove() throws EmptyQueueException;
    public void removeAll(A x);
}

public class Queue<A> implements API<A>{
     private A top;
     private Queue<A> rest;
     /*
      AF(c) = ...
      I(c) = ...
     */
     /* definizione costruttore e altri metodi omesse */
     public void add(A x)...
     public ... clone() ...
}

public class QueueS<A> extends Queue<A>{
     ....
     /*
      AF(c) =...
      I(c) = ...
     */
     /* costruttori */
     ...
     /* definizione altri metodi omessa */
     public void addAll(Vector<A> els) ...
}



/* Nota sul comportamento atteso per le operazioni di API
get: calcola primo in coda, se non vuota.
add: aggiunge, come ultimo della coda, l'elemento dato
addAll: aggiunge nell'ordine dato, come ultimi, gli elementi del
vettore.
remove: rimuove primo elemento.
removeAll: rimuove il primo elemento ed ogni sua copia.
*/
```