Linguaggi di Programmazione con Laboratorio
Corso di Laurea in Matematica
Appello del 18 Luglio 2018

Allegati

Esercizio 1.

/* NODE, EDGE, GRAPH, PATH
Una sola eccezione, monadica, per tutti i moduli, definita esternamente ad essi.
mk è un costruttore, getV restituisce l'etichetta, toString e copy sono gli additional per la presentazione e la duplicazione (corrispettivo di clone in Java), eq è l'additional per l'uguaglianza (stutturale o nominale secondo il tipo). Le operazioni getS e getT restituiscono nodi source e target. addN, AddE aggiungono liste di nodi e archi rispettivamente. Infine isEin controlla inclusione di un arco e nodes restituisce tutti i nodi nel grafo.
*/

```
exception InvalidArgException of string

module type NODE =
 sig type 'a node
  val mk: 'a -> 'a node
  val getV: 'a node -> 'a
  val toString: 'a node -> ('a -> string) -> string
  val copy: 'a node -> 'a node
  val eq: 'a node -> 'a node -> bool
end;;

module type EDGE =
 sig type ('a,'b) edge
  val mk: 'a Node.node -> 'a Node.node -> 'b -> ('a,'b)edge
  val getV: ('a,'b)edge -> 'b
  val getS: ('a,'b)edge -> 'a Node.node
  val getT: ('a,'b)edge -> 'a Node.node
  val toString:('a,'b)edge -> ('a->string) -> ('b->string) -> string
  val copy: ('a,'b)edge -> ('a,'b)edge
  val eq: ('a,'b)edge -> ('a,'b)edge -> bool
end;;

module type GRAPH =
 sig type ('a,'b)graph
  val mk: unit -> ('a,'b)graph
  val addN: 'a Node.node list -> ('a,'b)graph -> ('a,'b)graph
  val addE: ('a,'b)Edge.edge list -> ('a,'b) graph -> ('a,'b) graph
  val isEin: ('a,'b)Edge.edge -> ('a,'b) graph -> bool
  val nodes: ('a,'b) graph -> 'a Node.node list
  val toString:('a,'b)graph -> ('a->string) -> ('b->string) -> string
  val copy: ('a,'b)graph -> ('a,'b)graph
  val eq: ('a,'b)graph -> ('a,'b)graph -> bool
end;;
```