PROGRAMMAZIONE II (A,B) - a.a. 2017-18
Appello Straordinario – 10 aprile 2018

Esercizio 1

Si consideri il tipo di dati astratto modificabile `Container<T>`. Assumiamo che `Container<T>` sia utilizzato per memorizzare e reperire oggetti appartenenti alla classe `DataObject<T>` definita nel modo seguente

```
class DataObject<T> {
      private String nome;
      private T info;
     // opportuni metodi pubblici per poter
     // accedere e modificare i campi nome e info
}
```

Ogni oggetto inserito nello `Container<T>` ha associato un valore intero *versione*, rappresentata da un intero. Quando un oggetto di tipo `DataObject<T>` viene inserito nel `Container<T>`, se non ci sono oggetti con lo stesso nome gli viene assegnata la versione 0; altrimenti gli viene assengnata la massima versione esistente per oggetti con quel nome, più uno. Supponiamo che `Container<T>` fornisca le seguenti operazioni

- `put(DataObject<T> dataObj)`: inserisce l'oggetto associandogli la versione come descritto sopra, e restituisce la versione associata
- `get(String name, int version)`: restituisce il `DataObject<T>` di nome `name` e della versione richiesta, se esiste nel `Container<T>`

1. Si completi la specifica di `Container<T>` (overview con descrizione di un'istanza tipica e specifica completa dei metodi, compreso il tipo del risultato ed eventuali eccezioni lanciate).

   Si veda il file `Container.java`.

2. Si definisca una implementazione del tipo di dati astratto `Containet<T>` utilizzando come struttura di implementazione

   - `Vector <DataObject<T>> dataCollection`, contenente oggetti della classe `DataObject<T>`
   - `Vector <Integer> versioni`, contenente interi che rappresentano la versione del corrispondente `DataObject<T>`

3. Si forniscano la funzione di astrazione e l'invariante di rappresentazione e si dimostri che l'implementazione del metodo `put` preserva tale invariante.

   Si veda il file `MyContainer.java`.

Esercizio 2

Si estenda il linguaggio didattico funzionale in modo da includere il costrutto `SeqExp` per la definizione di sequenze di espressioni.

1. Si mostri come deve essere modificato l'interprete del linguaggio didattico funzionale.

   Si presenta una soluzione minimale che richiama l'esercizio `CodaLimitata` del febbraio scorso.

```
type exp = ...
   | SeqExp of seq
   ...
and seq = Empty | Item of exp * seq

type evT = ...
   | SeqExpVal of evT list

let rec eval (e : exp) (r : evT env) : evT = match e with
   ...
   | SeqExp e1 ->
        let s = (evalS q r) in SeqExpVal s
   ...
and let rec evalSeq (s : seq) (r : evT env) : evT list = match s with
     Empty -> []
   | Item (e, s1) -> (eval e r)::(evalSeq s1 r)
```

Esercizio 3

Si consideri il seguente programma OCaML

```
let c = 5;;
let first lst = match lst with
  | [] -> c
  | a::r -> a+5;;
let rest lst = match lst with
  | [] -> []
  | a::r -> r;;
let rec fr f lst z =
  if lst = [] then z
  else (f (first lst) (fr f (rest lst) z));;

fr (+) [2;3;4] 0;;
```

1. Si determini il tipo inferito dall'interprete OCaML per gli identificatori (`first`, `rest`, `fr`).

```
val first : int list -> int = <fun>
val rest : 'a list -> 'a list = <fun>
val fr : (int -> 'a -> 'a) -> int list -> 'a -> 'a = <fun>
```

2. Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.
   Si veda in fondo al testo.

3. Si determini il valore restituito dal programma.

```
- : int = 24
```

A

SL0	CL 0
c	5

B

SL A	CL A
first	F0

C

SL B	CL B
rest	F1

D

SL C	CL C
fr	F2

F0	code_first	A
F1	code_rest	B
F2	code_fr	D
F3	code_+	D

E

	SL A	CL D	
fr	f	F3	
(+)	lst	[2;3;4]	
[2;3;4]	z	0	
0	first lst	7	*
	rst lst	[3;4]	**
	fr f(rest lst) z)	17	####
f(first lst)(fr...)	res	24	

F

	SL A	CL E	
first	lst	[2;3;4]	POPPED
lst	res	7	*
a+5			

G

	SL B	CL E	
rest	lst	[2;3;4]	POPPED
lst	res	[3;4]	**
r			

H

	SL D	CL E	
fr	f	F3	
f	lst	[3;4]	
rest lst	z	0	
z	first lst	8	***
	rest lst	[4]	****
	fr f(rest lst) z)	9	###
f(first lst)(fr...)	res	17	####

I

	SL A	CL H	
first	lst	[3;4]	POPPED
lst	res	8	***
a+5			

L

	SL B	CL H	
rest	lst	[3;4]	POPPED
lst	res	[4]	****
r			

M

	SL D	CL H	
fr	f	F3	
f	lst	[4]	
rest lst	first lst	0	*****
z	rest lst	[]	#
	fr f(rest lst)z)	0	##
f(first lst)(fr...)	res	9	###

N

	SL A	CL M	
first	lst	[4]	POPPED
lst	res	9	*****
a+5			

O

	SL B	CL M	
rest	lst	[4]	POPPED
lst	res	[]	#

P

	SL D	CL M	
fr	f	F3	
f	lst	[]	
rest lst	z	0	
z	res	0	##