PROGRAMMAZIONE II (A,B) - a.a. 2018-19
Terzo Appello – 25 Giugno 2019

Domande di base

1. Si descriva brevemente il meccanismo di caricamento dinamico delle classi in Java.
2. Cosa è una *chiusura* e quale è il ruolo delle chiusure nella implementazione del linguaggi di programmazione.
3. Descrivere il modo in cui gli algoritmi di garbage collection *mark and sweep* e *reference counting* trattano strutture dati circolari allocate sullo heap.

Esercizio 1

Si consideri un tipo di dato astratto *HDSet*, *History Dependent Set* che rappresenta un insieme che memorizza anche tutti gli elementi che ha contenuto, anche quelli eventualmente rimossi. La classe `HDSet<E>` è generica rispetto al tipo `E` deli elementi. La classe `HDSet<E>` deve includere, tra gli altri, i seguenti metodi

- `public boolean contained(E el)` il cui effetto è quello di verificare se `el` è fra gli elementi che sono stati contenuti nell'insieme, inclusi quelli che lo sono attualmente.
- `public boolean newInsert(E el)` il cui effetto è quello di inserire `el` nell'insieme se `el` non appartiene all'insieme e non è fra gli elementi che sono stati contenuti.
- `public boolean wasContained(E el)` il cui effetto è quello di verificare se `el` è fra gli elementi che sono stati contenuti nell'insieme, ma non lo sono attualmente.
- `public boolean remove(E el)` il cui effetto è quello di rimuovere `el` nel caso sia presente fra gli elementi che sono attualmente contenuti nell'insieme.

1. Si definisca la specifica dei metodi descritti in precedenza, indicando per ogni metodo **a)** le clausole REQUIRES, MODIFY ed EFFECT, e **b)** il valore restituito e le eventuali eccezioni lanciate in dipendenza dei parametri attuali.
2. Si assuma di implementare la classe `HDSet<E>` con la struttura dati

```
private int size;
private ArrayList<E> elts;
private ArrayList<Boolean> presences; // altri parametri
public HDSet<E>( ) { // parametri ignorati
   size = 0;
   elts = new ArrayList<E>();
   presences = new ArrayList<Boolean>();
}
```

   - Si definiscano la funzione di astrazione e l'invariante di rappresentazione.
   - Si implementi il metodo `newInsert` verificando che preservi l'invariante di rappresentazione.

Esercizio 2

Si estenda il linguaggio didattico funzionale con il costrutto `StaticFun of ide * exp` per introdurre funzioni non ricorsive *statiche*. Una funzione statica è una funzione che non elimina l'ambiente locale della funzione: l'ambiente locale viene preservato per l' invocazione successiva. Si noti che l'espressione `exp` è definita da una sequenza di costrutti *let* annidati e dal corpo della funzione.

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

Esercizio 3

Si consideri il seguente programma OCaML

```
let z =2;;

let f1 = fun x y  -> x + y *z;;

let rec apply_n_times f n x =
  if n<=0 then x
  else apply_n_times f (n-1) (f x);;

let rec map_n_times g n = function
  | [] -> []
  | h1::ls -> (apply_n_times g n h1) :: map_n_times g n ls;;

let z = 3;;

let ff = f1 1;;

map_n_times ff z [10;20];;
```

1. Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.
2. Si determini il valore calcolato dal programma.