PROGRAMMAZIONE II (A,B) - a.a. 2018-19
Secondo Appello – 13 Febbraio 2019

Domande di base

1. Si consideri il seguente frammento di programma scritto in una sintassi simile al C.

```
int x;

int main() {
   x = 14;
   f();
   g();
}

void f() {
   int x = 13;
   h();
}

void g() {
   int x = 12;
   h();
}

void h() {
   printf("%d\n",x);
}
```

   Si descriva il risultato osservabile dell'esecuzione del programma in caso di (*i*) scoping statico e (*ii*) scoping dinamico. Si motivi la risposta.

2. Nell'ipotesi in cui l'istruzione `MyClass mc = new MyClass()` sia la prima istruzione del metodo `main`, si descriva brevemente le azioni effettuate dalla `JVM` durante l'esecuzione dell'istruzione precedente.

3. Descrivere il ruolo e le caratteristiche del principio di sostituzione nella realizzazione di applicazioni Java.

Esercizio 1

Si consideri il tipo di dato *IndexedOrderedList* che è un contenitore di dati generici. Un `IndexedOrderedList` è una collezione finita di liste di uno specifico tipo generico. Le liste della collezione sono *indicizzate* da un intero che permette di individuare in modo univoco le liste presenti nella collezione. Inoltre si assume che le liste siano ordinate in modo decrescente. Supponiamo che l'*interfaccia* del tipo di dato `IndexedOrderedList` sia definita nel modo seguente:

```
interface IndexedOrderedList<E> {
  /* Overview: Tipo modificabile di liste generiche, ordinate in modo decrescente,
      indicizzate da un valore intero */
  /* Typical element 0::L0, 1::L1, .... k::Lk,
     Li lista di elementi di tipo E  ordinata in modo decrescente */

   /* Restituisce la lista di indice index nella collezione */
   List<E>  search(Integer index);

   /* Restituisice l'indice della lista elts se presente nella collezione. */
   Integer  indexOf(List<E> elts);
```

```
/* Inserisce nella lista  individuata dall'indice index  l'elemento elem */
void put(E elem, Integer index);

/* altri metodi */
}
```

1. Assumendo di adottare una strategia di programmazione difensiva, si completi il progetto del tipo di dato astratto `IndexedOrderedList<E>`, definendo le clausole REQUIRES, MODIFIES e EFFECTS di ogni metodo, indicando le eccezioni eventualmente lanciate e se sono checked o unchecked.
2. Si consideri la seguente struttura di implementazione per la classe `MyIndexedOrderedList<E>`

   ```
   private HashMap<Integer, ArrayList<E>> hmap;
   private int size;
   ```

   Si definisca l'invariante di rappresentazione per l'implementazione di `MyIndexedOrderedList<E>`.
3. Si fornisca l'implementazione del costruttore e dei metodi `search` e `put` e si dimostri che le implementazioni proposte preservano l'invariante di rappresentazione.

Esercizio 2

Si estenda il linguaggio didattico funzionale con i costrutti opportuni per poter operare con `Stack` contenenti valori interi.

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

Esercizio 3

Si consideri il seguente programma OCaml

```
let const = 10;;

let myfun l x y = match l with
  [] -> x
  | a::ls -> if a > 0 then x+y else y-x;;

 let checkAndApply l (f:int ->int) =
  let rec aux l f = match l with
    [] -> const
   | elem::ls -> if elem > 0 then f const else aux ls f
   in aux l f;;

let ls1 = [100;1000];;

let ls2 = [1;2;3;4];;

let const = 5;;

checkAndApply ls2 ((myfun ls1) const);;
```

1. Si mostri la struttura della pila dei record di attivazione al momento dell'invocazione della funzione identificata dal parametro formale `f`.
2. Si determini il valore calcolato dal programma.