PROGRAMMAZIONE II (A,B) - a.a. 2019-20
Seconda Valutazione Intermedia – 19 Dicembre 2019

Domande di base

- Si consideri il seguente programma Java

```
class A {
    public void foo (Object o) { System.out.println("A"); }
}
class B {
public void foo (String o) { System.out.println("B"); }
}

class C extends A {
public void foo (String s) { System.out.println("C"); }
}
class D extends B {
public void foo (Object o) { System.out.println("D"); }
}

class Main {
public static void main(String[] args) {
   A a = new C(); a.foo("Java");
   C c = new C(); c.foo("Java");
   D d = new D(); d.foo("Java");
   B b = new D(); b.foo("Java");
   }
}
```

  1. Si descriva la struttura della tabella dei metodi (*vtable*) per tutte le classi presenti nel programma. Motivare la risposta.
  2. Si descriva l'ordine in cui sono caricate le classi durante l'esecuzione del metodo `main`. Motivare la risposta.

- Spiegare il ruolo del meccanismo della *retention* nell'implementazione dei linguaggi funzionali.
- Descrivere l'impatto delle regole di *scope* nell'implementazione dei linguaggi di programmazione.
- Tutte le variabili locali del metodo `main` di una applicazione Java appartengono al *rootset*? Si motivi la risposta.

Esercizio 1

Si consideri il seguente programma OCAML

```
let length list =
    let rec aux n = function
      | [] -> n
      | _::t -> aux (n+1) t
    in aux 0 list;;
let apply = fun a -> ( fun b -> a b );;
let k = 100;;
let f = fun x -> x+k;;
let g = fun x -> x-k;;
let k = 150;;
let l = [10;15;20];;
apply (if (length l  > 2) then g else f ) k;;
```

1. Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.
2. Assumendo di adottare una regola di scoping dinamico, si descriva quali sono le modifiche che avvengono nella struttura a run-time dei record di attivazione durante l'esecuzione del programma.

Esercizio 2

Estendere il linguaggio didattico funzionale con il tipo di dato `IntSet` per poter operare con insiemi finiti di valori interi mediante un insieme di operatori primitivi quale `create, isEmpy, insert, remove, isIn` con il significato ovvio.

1. Definire le regole di valutazione delle operazioni primitive su insiemi di valori interi e estendere consistentemente la struttura dell'interprete del linguaggio didattico funzionale.
2. Introdurre una nuova tipologia di astrazione funzionale che consente di applicare una funzione a tutti gli elementi di un insieme di valori interi. Assumiamo per semplicità che l'astrazione introdotta non sia ricorsiva. Supponiamo inoltre, che la sintassi concreta dell'astrazione funzionale abbia la forma `FunOverSet(x:int, s: set) = body`. Mostrare come deve essere modificato l'interprete del linguaggio didattico per includere questa nuova forma di astrazione funzionale.