PROGRAMMAZIONE II (A,B) - a.a. 2014-15
Seconda Valutazione Intermedia — 29 maggio 2015

Esercizio 1. Si consideri il seguente programma a oggetti scritto in una sintassi Java-like.

```
class B { ... }

class A {
   static B b1;
   public B b2;
}

public class Tester {

   public static void main(String[] args) {
      B b1 = new B();
      B b2 = new B();
      A a1 = new A();
      A a2 = new A();

      a1.b1 = b1;
      a1.b2 = b1;
      a2.b2 = b2;

      a1 = null;
      b1 = null;
      b2 = null;
     (1)
     // altro codice
  }
}
```

1. Si descrivano le informazioni presenti a run-time al termine dell'esecuzione del metodo main nel punto (1), e si mostri quali sono gli oggetti allocati sullo heap che sono divenuti garbage.

```
Solamente l'oggetto inizialmente tiferito da a1 diventa garbage.
```

Esercizio 2. Si consideri il seguente programma scritto in OCaml.

```
let k = 5 in
    let pick_one n = if n < 0 then (fun x -> -x)
                     else if n = 0 then (fun x -> k)
                          else (fun x -> x + k) in
        let rec twice_app (f, l) = match l with
                                       [] -> []
                                     | (h::r) -> ((f h) h) :: twice_app(f, r) in
            twice_app (pick_one, [-9; 2; 0]);;
```

1. Si determini il tipo di tutti gli identificatori che compaiono nel programma scelto.

```
val k : int = 5
val pick_one : int -> int -> int = <fun>
val twice_app : ('a -> 'a -> 'b) * 'a list -> 'b list = <fun>
```

2. Si descriva lo stato dello stack dei record di attivazione subito dopo l'invocazione della funzione `twice_app`. Si indichi per ogni record di attivazione le informazioni relative al puntatore di catena statica, puntatore di catena dinamica e struttura dell'ambiente locale.

3. Qual è il valore calcolato dal programma? `- : int list = [9; 7; 5]`

Esercizio 3. Si consideri il linguaggio didattico funzionale, e se ne estenda la sintassi in modo da includere la possibilità di assegnamenti multipli. Le espressioni da assegnare devono essere valutate tutte assieme, e si assume che nel caso di più assegnamenti alla stessa variabile valga quello più a destra. Si veda il seguente esempio, espresso in una sintassi OCaml-like.

```
let x = 5 and y = 10 in x + y ;;
> val - : int = 15

let x = 5 and x = 10 in x + x ;;
> val - : int = 20
```

1. Si estenda la sintassi astratta del linguaggio didattico funzionale in modo da includere espressioni con assegnamenti multipli.

```
type exp = . . .
         | MLet of (ide * exp) list * exp
```

2. Si estenda il codice OCaml dell'interprete per trattare la valutazione di tali espressioni.

```
let rec sem (e: exp) (r: eval env) = match e with
           . . .
         | MLet(d, b) -> let l = evallist(r, d) in
                           let r1 = bindlist(r, l) in sem(b, r1)

let rec evallist (r: eval env) (d: (ide*exp) list) = match d with
           [] -> []
         | (i, e) :: d1 -> let v = sem(e, r) in (i, v) :: evallist(r, d1)
```

```
let rec sem (e: exp) (r: eval env) = match e with
           . . .
         | MLet(d, b) -> let r1 = newbindlist(r, r, d) in sem(b, r1)

let rec newbindlist (r: eval env) (r1: eval env) (d: (ide*exp) list) = match d with
           [] -> r
         | (i, e) :: d1 -> let v = sem(e, r) in
                             let r2 = bind(r1, i, v) in newbindlist(r, r2, d1)
```

Esercizio 4. Si consideri il seguente programma scritto in una sintassi C-like.

```
int i = 2;
void foo(int f, int g) {
   f = f - i;
   g = f;
}

int main( ) {
   int a[] = {2, 0, 1};
   foo(i, a[i]);
   printf("%d %d %d %d\n", i, a[0], a[1], a[2]);
}
```

1. Quali sono i valori stampati dal programma nel caso di passaggio di parametri per

   (a) call-by-value, `2 2 0 1`

   (b) call-by-reference, `0 2 0 0`

   (c) call-by-name? `0 0 0 1`

   Si motivi la risposta.