PROGRAMMAZIONE II (A,B) - a.a. 2013-14
Appello Straordinario — 6 Novembre 2014

Esercizio 1. Si consideri il seguente programma OCaml:

```
let f x y z =
  if z > 0 then (fun w -> w + x + y)
                        else (fun w -> w + x - y);;
let a = 1;;
let b = 2;;
let c = f b a;;
let d = c 7;;
let res = d 4;;
```

1. Indicare il tipo inferito dall'interprete OCaml per le variabilli `f` e `c`.
2. Indicare il valore associato alla variabile `d` al termine dell'esecuzione del programma.
3. Simulando la valutazione del programma, mostrare la struttura della pila dei record di attivazione subito dopo l'invocazione di `let c = f b a`, e al momento dell'invocazione di `d 4`.
4. Assumendo di considerare una versione di OCaml con regole di scoping dinamico, mostrare la struttura della pila dei record di attivazione subito dopo l'invocazione di `let c = f b a`, e al momento dell'invocazione di `d 4`.

Esercizio 2. Si consideri un tipo di dato astratto *Membro* di supporto alla gestione di una rete sociale. La classe `Membro` ha, tra gli altri, i seguenti metodi:

1. `public void invita (Membro m)`

   il cui effetto é quello di invitare un altro membro della rete sociale a diventare amico del menro this. Il metodo non deve permettere di invitare se stessi oppure un membro che ha giŕisposto positivamente a un invito.

2. `public void accettaInvito (Membro m)`

   il cui effetto é quello di accettare l'invito di un altro membro della rete sociale a patto che questi abbia effettivamente invitato this.

3. `public void declinaInvito (Membro m)`

   il cui effetto é quello di rifiutare l'invito di un altro membro della rete sociale a this, rimuovendo anche il relativo invito.

1. Definire la specifica dei metodi descritti in precedenza e indicando per ogni metodo: **a)** le clausole REQUIRES, MODIFY ed EFFECT; **b)** il valore restituito e le eventuali eccezioni lanciate in dipendenza dei parametri attuali. Si ipotizzi che la classe `Membro` fornisca anche i seguenti metodi osservatori

   - `public ArrayList<Membro> invitati()`
     il cui effetto é quello di restituire l'elenco dei membri invitati da this che non hanno ancora risposto all'invito
   - `public ArrayList<Membro> amici()`
     il cui effetto é quello di restituire l'elenco dei membri invitati da this o che hanno invitato this (uno ha invitato l'altro e l'invito é stato accettato)

Si assuma di implementare la classe `Membro` con una struttura dati strutturata nel modo seguente:

```
private ArrayList<Membro> altriMembri;
private int inizioInviti;
public Membro( ...) {// parametri ignorati
  altriMembri = new ArrayList<Membro>;
  inizioInvitato 0 =;
  }
```

2. Definire l'invariante di rappresentazione.

3. Implementare i metodi `accettaInvito` verificando che preservi l'invariante di rappresentazione

Esercizio 3. Si consideri il linguaggio didattico imperativo. Estendiamo il linguaggio in modo da includere la possibilità di dichiarare procedure in cui i parametri siano passati per *value-result*. Per semplicità supponiamo che le procedure abbiamo un unico parametro formale.

1. Estendere la sintassi astratta del linguaggio didattico imperativo in modo da includere la dichiarazione di procedure con parametri per value-result e la loro invocazione.

2. Definire le regole OCaml dell'interprete per trattare la valutazione di dichiarazione di procedure con parametri per value-result e la loro invocazione.