Linguaggi e Programmazione Orientata agli Oggetti

Prova scritta

a.a. 2017/2018

11 luglio 2018

1. (a) Indicare quali delle asserzioni contenute nel seguente codice Java hanno successo e quali falliscono, motivando la risposta.

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherTest {
    public static void main(String[] args) {
        Pattern regEx = Pattern.compile("(zero|one)|(\\s+)|([A-Z]([a-zA-Z]*))");
        Matcher m = regEx.matcher("zero One");
        m.lookingAt();
        assert !(m.group(1) == null);
        assert !(m.group(0) == null);
        m.region(m.end(), m.regionEnd());
        m.lookingAt();
        assert m.group(1) == null;
        assert m.group(0) != null;
        m.region(m.end(), m.regionEnd());
        m.lookingAt();
        assert m.group(3).equals("One");
        assert m.group(4).equals("ne");
    }
}
```

   (b) Mostrare che la seguente grammatica è ambigua.

```
Exp  ::= ! Exp | Exp ∧ Exp | ( Exp ) | Bool
Bool ::= false | true
```

   (c) Modificare la grammatica definita al punto precedente in modo che **non sia ambigua** e che il linguaggio generato a partire dal non terminale `Exp` **resti invariato**.

2. Sia `merge : ('a * 'b -> 'c) -> ('a * 'b) list -> 'c list` la funzione così specificata: `merge` f $[(a_1, b_1); \ldots; (a_n, b_n)]$ restituisce la lista $[f(a_1, b_1); \ldots; f(a_n, b_n)]$.
   Esempio:

```
# merge (fun (x,y) -> x+y) [(1,2);(3,4);(5,6)];;
- : int list = [3; 7; 11]
# merge (fun (x,y) -> x+String.length y)  [(1,"one");(2,"two");(3,"three")];;
- : int list = [4; 5; 8]
```

   (a) Definire `merge` senza uso di parametri di accumulazione.
   (b) Definire `merge` usando un parametro di accumulazione affinché la ricorsione sia di coda.
   (c) Definire `merge` come specializzazione della funzione `map:('a -> 'b) -> 'a list -> 'b list`.

3. Completare la seguente classe `OddOnlyIterator` di iteratori definiti a partire da un iteratore di base `baseIterator` che restituiscono solo gli elementi di posizione dispari di `baseIterator` (considerando dispari la posizione del primo elemento).

```
public class OddOnlyIterator<E> implements Iterator<E> {

    private final Iterator<E> baseIterator; // non opzionale
    private boolean returnNext = true; /* stabilisce se il prossimo elemento di
        baseIterator va restituito; inizialmente true poiche' il primo
        elemento di baseIterator e' in posizione dispari */

    public OddOnlyIterator(Iterator<E> baseIterator) { /* completare */ }
    public boolean hasNext() { /* completare */ }
    public E next() { /* completare */ }
}
```

Per esempio, nel codice sottostante viene creato l'iteratore `altIt` a partire dall'iteratore `it` della lista `[2, 4, 6, 8]` e l'iterazione su `altIt` restituisce solo gli elementi 2 e 6.

```
Iterator<Integer> it = asList(2, 4, 6, 8).iterator();
Iterator<Integer> altIt = new OddOnlyIterator<>(it);
while (altIt.hasNext())
    System.out.println(altIt.next()); // stampa 2\n6\n
```

4. Considerare le seguenti dichiarazioni di classi Java:

```
public class P {
    String m(long l) { return "P.m(long)"; }
    String m(int i) { return "P.m(int)"; }
}
public class H extends P {
    String m(long l) { return super.m(l) + " H.m(long)"; }
    String m(Integer i) { return super.m(i) + " H.m(Integer)"; }
}
public class Test {
    public static void main(String[] args) {
        P p = new P();
        H h = new H();
        P p2 = h;
        System.out.println(...);
    }
}
```

Dire, per ognuno dei casi elencati sotto, che cosa succede sostituendo al posto dei puntini nella classe `Test` il codice indicato, assumendo che tutte le classi siano dichiarate nello stesso package.

Per ogni caso fornire due o tre righe di spiegazione così strutturate: se c'è un errore in fase di compilazione, specificare esattamente quale; se invece la compilazione va a buon fine spiegare brevemente perché e descrivere cosa avviene al momento dell'esecuzione, anche qui spiegando brevemente perché.

(a) `p.m(42)`

(b) `p2.m(42)`

(c) `h.m(42)`

(d) `p.m(42L)`

(e) `p2.m(42L)`

(f) `h.m(42L)`