Linguaggi e Programmazione Orientata agli Oggetti

Prova scritta parziale

a.a. 2018/2019

11 febbraio 2019

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

```
Pattern regEx = Pattern.compile("([A-Z][_a-zA-Z]*)|(int|bool|double)|(\\s+)");
Matcher m = regEx.matcher("Is_bool double");
m.lookingAt();
assert m.group(2) == null;
assert m.group(0).equals("Is_bool");
m.region(m.end(), m.regionEnd());
m.lookingAt();
assert m.group(2) == null;
assert m.group(3) != null;
m.region(m.end(), m.regionEnd());
m.lookingAt();
assert m.group(2) != null;
assert m.group(0).equals("double");
```

   (b) Mostrare che la seguente grammatica è ambigua.

```
Exp  ::= Exp ! | Exp * Exp | < Exp > | Id
Id ::= x | y
```

   (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 `cond_map : ('a -> 'b) -> ('a -> 'b) -> ('a -> bool) -> 'a list -> 'b list` la funzione così specificata:

   `cond_map` f g p l restituisce la lista ottenuta da l applicando, nell'ordine, la funzione f agli elementi di l che soddisfano il predicato p e la funzione g a quelli che non lo soddisfano.

   Esempio:

```
# cond_map  sqrt (fun x -> 0.) (fun x -> x>=0.0) [-1.0;9.0;-4.0;4.0]
- : float list = [0.; 3.; 0.; 2.]
```

   (a) Definire `cond_map` senza uso di parametri di accumulazione.

   (b) Definire `cond_map` usando un parametro di accumulazione affinché la ricorsione sia di coda.

   (c) Definire `cond_map` come specializzazione della funzione `List.map : ('a -> 'b) -> 'a list -> 'b list .`

3. Considerare le seguenti dichiarazioni di classi Java:

```
public class P {
   String m(Object o) { return "P.m(Object)"; }
   String m(Number n) { return "P.m(Number)"; }
}
public class H extends P {
   String m(int i) { return super.m(i) + " H.m(int)"; }
}
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(42.0)`

(e) `p2.m(42.0)`

(f) `h.m(42.0)`