Linguaggi e Programmazione Orientata agli Oggetti

Prova scritta parziale

18 febbraio 2020, a.a. 2019/2020

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("(\\s+)|([01]+)|([a-z][a-zA-Z0-9]*)|(ADD|SUB|MUL|DIV)");
      Matcher m = regEx.matcher("ADD1010x01");
      m.lookingAt();
      assert "".equals(m.group(1));
      assert "ADD".equals(m.group(4));
      m.region(m.end(), m.regionEnd());
      m.lookingAt();
      assert "1010".equals(m.group(0));
      assert null == m.group(2);
      m.region(m.end(), m.regionEnd());
      m.lookingAt();
      assert "x01".equals(m.group(3));
      assert null == m.group(2);
   }
}
```

   (b) Mostrare che nella seguente grammatica la stringa `c-d/c` è ambigua rispetto a `Exp`.

```
Exp ::= Exp / Base | Exp - Base | Base
Base ::=  Id | { Exp } | Exp
Id ::= c | d
```

   (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 `unzip : ('a * 'b) list -> 'a list` la funzione così specificata:

   `unzip` $[(x_1, y_1); \ldots; (x_n, y_n)]$ restituisce la lista $[x_1; \ldots; x_n]$.

   Esempi:

```
unzip [(1, "one"); (2, "two"); (3, "three")] = [1; 2; 3];;
unzip [("one", 1); ("two", 2); ("three", 3)] = ["one"; "two"; "three"];;
```

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

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

   (c) Definire `unzip` 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(double d) {
      return "P.m(double)";
   }
   String m(double[] da) {
      return "P.m(double[])";
   }
}
public class H extends P {
   String m(float f) {
      return super.m(f) + " H.m(float)";
   }
   String m(float[] fa) {
      return super.m(new double[]{ fa[0], fa[1] }) + " H.m(float[])";
   }
}
public class Test {
   public static void main(String[] args) {
      P p = new P();
      H h = new H();
      P p2 = h;
      float f = 4.2f;
      double[] da = { 1.2, 2.3 };
      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(f)` (b) `p2.m(f)` (c) `h.m(f)` (d) `p.m(da)` (e) `p2.m(da)` (f) `h.m(da)`