Linguaggi e Programmazione Orientata agli Oggetti

Prova scritta

a.a. 2020/2021

21 giugno 2021

1. (a) Per ogni stringa elencata sotto stabilire, motivando la risposta, se appartiene alla seguente espressione regolare e, in caso affermativo, indicare il gruppo di appartenenza (escludendo il gruppo 0).

```
(\s+)|(0[0-7]*)|(0x1|0x2|0x3)|([0-9][a-zA-Z]+)
```

   i. `"0x4"`
   ii. `"0 0"`
   iii. `"0x"`
   iv. `"0x1"`
   v. `"001"`
   vi. `"0a"`

   (b) Mostrare che la seguente grammatica è ambigua.

```
Exp ::= Exp and Atom | First Atom | Atom
First ::= Exp and
Atom ::= false | true | ( Exp )
```

   (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 `cat : (string * string) list -> string list` la funzione così specificata:

   `cat [`(x_1, y_1)`; ... ;`(x_k, y_k)`] = [`x_1 `^` y_1`; ... ;`x_k `^` y_k`]`, dove $k \geq 0$ e `^` rappresenta l'operatore di concatenazione tra stringhe in OCaml.

   Esempi:

```
cat [("hello"," world");("ciao ","mondo")]=["hello world"; "ciao mondo"]
cat []=[]
```

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

   (b) Definire `cat` usando `List.map:('a -> 'b) -> 'a list -> 'b list`.

3. Completare la seguente classe di iteratori `CharSeqIterator` per iterare, nell'ordine convenzionale dal primo all'ultimo elemento, sui caratteri di una sequenza di tipo `java.lang.CharSequence`.

   Esempio:

```
for (var ch : new CharSeqIterator("abc")) System.out.println(ch); // prints a b c
for (var ch : new CharSeqIterator("")) System.out.println(ch); // no printed chars
for (var ch : new CharSeqIterator("aBc")) System.out.println(ch); // prints a B c
```

   L'interfaccia predefinita `java.lang.CharSequence` (implementata da `java.lang.String`) contiene, tra gli altri, i metodi `char charAt(int index)` e `int length()`: il primo restituisce il carattere della sequenza che si trova all'indice `index` (gli indici partono da zero), il secondo calcola la lunghezza della sequenza.

Codice da completare:

```
import java.util.Iterator;
import java.util.NoSuchElementException;
import static java.util.Objects.requireNonNull;

class CharSeqIterator implements Iterator<Character>, Iterable<Character> {

   // dichiarare i campi mancanti
   // invariant charSeq!=null

   public CharSeqIterator(CharSequence charSeq) {
      // completare
   }

   @Override
   public boolean hasNext() {
      // completare
   }

   @Override
   public Character next() {
      // completare
   }

   @Override
   public Iterator<Character> iterator() { return this; }
}
```

4. Considerare le seguenti dichiarazioni di classi Java:

```
public class P {
   String m(double d) { return "P.m(double)"; }
   String m(float f) { return "P.m(float)"; }
}
public class H extends P {
   String m(long l) { return "H.m(long)"; }
   String m(int i) { return "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)`