# Linguaggi per il Web – appello del 25/6/2013

COGNOME: _____

NOME: _____

MATRICOLA: _____

Autorizzo la pubblicazione del mio voto di questo esame sul sito web `http://www.dis.uniroma1.it/~rosati/lw`, secondo quanto prevede il decreto legislativo 196/2003 (codice in materia di protezione dei dati personali) che dichiaro di conoscere. In fede,

_____

**Esercizio 1** (a) Data la seguente grammatica $G$:

$$S \to CS \mid BA$$
$$A \to cA \mid b$$
$$B \to baB \mid cSCB \mid aA$$
$$C \to abBC \mid bBA \mid cBA$$

è possibile stabilire se $G$ è una grammatica LL(1) senza costruire esplicitamente gli insiemi FIRST e FOLLOW e la tabella di parsing di $G$? motivare la risposta;

(b) Data la seguente grammatica $G$:

$$S \to Sa \mid a \mid SYb \mid SYZc$$
$$X \to XZY \mid XZZ \mid ab$$
$$Y \to abY \mid aZX \mid abcX$$
$$Z \to Y \mid X \mid YZ \mid XZ$$

scrivere una grammatica $G'$ tale che $G'$ non presenti né ricorsione sinistra diretta né prefissi comuni e tale che $\mathcal{L}(G') = \mathcal{L}(G)$.

**Esercizio 2** Si consideri il frammento del linguaggio Java costituito dalle stringhe che corrispondono alla definizione di una sequenza di istruzioni Java. Una sequenza di istruzioni è delimitata da parentesi graffe e contiene zero o più istruzioni. Ogni istruzione può essere: (i) una dichiarazione di una o pi ù variabili (separate da virgole) di tipo `int` o `float`; (ii) una assegnazione, il cui lato destro può essere una costante intera, una costante di tipo `float`, un identificatore di variabile, o una somma di sottoespressioni di questo tipo; (iii) una istruzione di tipo `if`, con ramo `else` opzionale, il cui ramo `then` e l'eventuale ramo `else` contengono una istruzione, e la cui condizione è un confronto di uguaglianza o disuguaglianza tra due espressioni, ognuna delle quali può essere una costante intera, una costante di tipo `float`, un identificatore di variabile, o una somma di sottoespressioni di questo tipo; (iv) una istruzione `while` il cui corpo contiene una istruzione e la cui condizione è un confronto di uguaglianza o disuguaglianza tra due espressioni, ognuna delle quali può essere una costante intera, una costante di tipo `float`, un identificatore di variabile, o una somma di sottoespressioni di questo tipo; (v) una sequenza di istruzioni delimitata da parentesi graffe.

Esempi di stringhe appartenenti a questo linguaggio sono i seguenti:

```
{
  int x,y,z;
  z=1;
  y=54+z+100;
  if (z==y+1000) x=12+y;
}
```

```
{
  paperino=-3.24E-6;
  z=met4+1+x;
  while (x+y!=15+z+3) a=2;
  if (x==0) {
    int w,p1;
    w=1+z;
  }
  else b=c;
  { z=1; x=2; int w,f; }
  { z=1; x=2; { } y=3+c; }
}
```

```
{
  float paperino, a, b;
  int x,y,z;
  paperino=2.25E18+.3;
  while (x+y!=15+z+3) {
    a=2;
    while (c==d+1.1) { b=c; }
    if (x==0) {
      if (x!=y+3+z) w=1+z;
      else { p=-4.5E-15+1.0; x=0; }
    }
  }
  else b=c;
}
```

Scrivere una grammatica non contestuale per tale linguaggio, dividendo la specifica del lessico del linguaggio (che va definita mediante espressioni regolari) dalla specifica della sintassi vera e propria. Scrivere preferibilmente tale specifica come specifica JavaCC.

**Esercizio 3** Data la seguente DTD:

```
<!DOCTYPE r [
  <!ELEMENT r ((y|z),(x|y))>
  <!ELEMENT x ((w1|w2)*,(w3,w4)+)>
  <!ELEMENT y ((w3|w4)*,(w1,w2)+)>
  <!ELEMENT z (w1?,w2*,w3?)>
  <!ELEMENT w1 (#PCDATA|r)*>
  <!ELEMENT w2 (#PCDATA)>
  <!ELEMENT w3 EMPTY>
  <!ELEMENT w4 (#PCDATA|x)>
  <!ATTLIST x attrx CDATA #IMPLIED>
  <!ATTLIST y attry CDATA #REQUIRED>
  <!ATTLIST z attrz CDATA #REQUIRED>
]>
```

1) dire se la DTD è corretta ed in caso negativo evidenziare gli errori presenti e correggerli; 2) scrivere un documento XML che sia valido rispetto alla DTD (eventualmente corretta).

**Esercizio 4** Data la seguente DTD:

```
<!DOCTYPE r [
  <!ELEMENT r (y,x)*>
  <!ELEMENT y ((z1|z2)+,(z3,z4)*)>
  <!ELEMENT x (z1?,z2,z3*)>
  <!ELEMENT z1 (#PCDATA)>
  <!ELEMENT z2 (#PCDATA)>
  <!ELEMENT z3 (#PCDATA)>
  <!ELEMENT z4 EMPTY>
  <!ATTLIST x attrx CDATA #IMPLIED>
  <!ATTLIST y attry CDATA #REQUIRED>
]>
```

scrivere un XML Schema corrispondente a tale DTD.

**Esercizio 5** Scrivere un foglio di stile XSL che, dato un documento XML, restituisce il documento tale che: 1) l'elemento radice di input viene copiato e il suo contenuto viene ricorsivamente trasformato; 2) ogni elemento figlio dell'elemento radice di input viene cancellato, e il suo contenuto viene ricorsivamente trasformato; 3) ogni elemento che è figlio di un figlio dell'elemento radice viene copiato in output come figlio dell'elemento radice di output, aggiungendo come sottoelemento un elemento vuoto `figlio-figlio-radice`. Inoltre, il contenuto dell'elemento di input viene ricorsivamente trasformato; 4)tutti gli altri elementi vengono cancellati e il loro contenuto viene ricorsivamente trasformato: 5) tutte le parti testuali del documento di input vengono copiate in output.

Ad esempio, se il documento XML di input è il seguente:

```
<r>
  <d>testo 0
     <rad>testo 1</rad>
  </d>
  <b>
    <p>
      <w>
        <z>testo 2</z>
      </w>
    </p>
  </b>
  <c>
    <p/>
    <y>testo 3
      <d>testo 4</d>
    </y>
  </c>
  <z/>
</r>
```

il foglio di stile applicato al documento deve restituire il documento seguente:

```
<r>
  testo 0
  <rad>
    <figlio-figlio-radice/>
    testo 1
  </rad>
  <p>
    <figlio-figlio-radice/>
    testo 2
  </p>
  <p>
    <figlio-figlio-radice/>
  </p>
  <y>
    <figlio-figlio-radice/>
    testo 3
    testo 4
  </y>
</r>
```

**Esercizio 6** Siano `b`, `c`, `d`, `r`, `s` e `t` delle URI di un namespace con prefisso `p`. Scrivere un modello RDF che rappresenta le seguenti informazioni: "Le URI `b`, `c`, `d`, `r`, `s` e `t` rappresentano persone i cui nomi sono, rispettivamente, Beatrice, Carlo, Dario, Renata, Simona, Tommaso. Inoltre: Tommaso conosce una persona; Dario e Renata conoscono una persona che conosce Simona; infine, Carlo conosce una persona che crede che Simona conosca Beatrice e Tommaso. Usare la URI `foaf:name` per esprimere il predicato "ha nome" e la URI `foaf:knows` per esprimere il predicato "conosce".