Linguaggi per il Web – appello del 24/9/2013

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:

$$S \to aS \mid bXX$$
$$X \to cd \mid bYX$$
$$Y \to ZZaaa$$
$$Z \to aXb \mid bYc \mid cZd \mid da$$

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:

$$S \to SSA \mid SSB \mid SSC \mid AS \mid A$$
$$A \to baS \mid cSCB \mid bab$$
$$B \to Dba \mid Dbc \mid Bdc \mid BS$$
$$C \to CcbaD \mid Cc \mid d$$
$$D \to DD \mid Ac \mid d \mid DDbb$$

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 zero o più istruzioni. Ogni istruzione può essere: (i) una dichiarazione di una o più variabili (separate da virgole) di tipo `int` o `float` o `String`; (ii) una assegnazione, il cui lato destro può essere una costante intera, una costante di tipo `float`, una costante di tipo stringa, un identificatore di variabile, una invocazione di metodo, o una somma di sottoespressioni di questo tipo; (iii) una invocazione di metodo, in cui ogni argomento (parametro attuale) può essere una costante intera, una costante di tipo `float`, una costante di tipo stringa, un identificatore di variabile, oppure una somma di sottoespressioni di questo tipo; (iv) 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`, una costante di tipo stringa, un identificatore di variabile, o una somma di sottoespressioni di questo tipo; (v) 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`, una costante di tipo stringa, un identificatore di variabile, o una somma di sottoespressioni di questo tipo; (vi) una sequenza (anche vuota) di istruzioni delimitata da parentesi graffe.

Esempi di stringhe appartenenti a questo linguaggio sono i seguenti:

```
z=-1;
x=metodo3(123+y);
metodo4();
{ y = p(0, "ciao"); }
while (a==-3.42)
  x23=-0.23E-12 + x;
```

```
paperino="ciao";
{ z=metodo5();
  {  y=met3(x+z,-1,3.0+12.4);
     z=metodo6("ciao",z,1); }
  if (p!="abc xyz" + q) a=1
  else if (x==0) { a="abcd"; } }
```

```
a123=z;
{  paperino=123;
   { x=met1(x,abc12,0,a,y); y="ciao"; } }
if (z==3.14) y="ok"
else {
  pluto(25,x+"abc");
  if (p4!=-0.32e-3) x=1.15; }
```

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 a [
  <!ELEMENT a (a*,(u|s|z),a*,y,((t,y)|(w,z)))>
  <!ELEMENT s (#PCDATA|w)*>
  <!ELEMENT t (x?,w*,(t|a|s)?)>
  <!ELEMENT u (a?,(u|s)*,(z,y)+)>
  <!ELEMENT x (#PCDATA|x)*>
  <!ELEMENT y (#PCDATA|x)*>
  <!ELEMENT w EMPTY>
  <!ELEMENT z ((z,y)*,(u|s)+)>
  <!ATTLIST s attrs CDATA #REQUIRED>
  <!ATTLIST u attru CDATA #IMPLIED>
  <!ATTLIST x attrx 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 (a|c)*>
  <!ELEMENT a ((d,b)+,(g|f)*)>
  <!ELEMENT b (#PCDATA)>
  <!ELEMENT c (d*,b?,g+)>
  <!ELEMENT d (#PCDATA)>
  <!ELEMENT f (#PCDATA)>
  <!ELEMENT g EMPTY>
  <!ATTLIST a attra CDATA #REQUIRED>
  <!ATTLIST c attrc CDATA #IMPLIED>
]>
```

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 trasformato in un elemento `r`, e il suo contenuto viene ricorsivamente trasformato; 2) per ogni elemento `x` che è figlio dell'elemento radice viene creato in output un elemento `nx`, e viene creato, come sottoelemento di tale elemento `nx`, un elemento vuoto `oldx`. Inoltre, il contenuto dell'elemento di input viene ricorsivamente trasformato; 3) ogni elemento `y` che è figlio dell'elemento radice non viene copiato, ma il suo contenuto viene ricorsivamente trasformato; 4) ogni altro elemento figlio dell'elemento radice di input viene copiato, e il suo contenuto viene ricorsivamente trasformato; 5) per ogni elemento che è figlio di un figlio dell'elemento radice viene creato in output un elemento `new`, e viene creato, come sottoelemento di tale elemento `new`, un elemento vuoto il cui nome è uguale al nome dell'elemento di input corrente. Inoltre, il contenuto dell'elemento di input viene ricorsivamente trasformato; 6) tutti gli altri elementi non vengono copiati e il loro contenuto viene ricorsivamente trasformato: 7) nessuna parte testuale del documento di input viene copiata in output.

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

```
<a>
  <x>testo 0
    <rad>testo 1</rad>
  </x>
  <b>
    <p>
      <w>
        <z>testo 2</z>
      </w>
    </p>
  </b>
  <y>
    <p/>
    <y>testo 3
      <d>testo 4</d>
    </y>
  </y>
  <f/>
</a>
```

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

```
<r>
  <nx>
    <oldx/>
    <new>
      <rad/>
    </new>
  </nx>
  <b>
    <new>
      <p/>
    </new>
  </b>
  <new>
    <p/>
  </new>
  <new>
    <y/>
  </new>
  <f/>
</r>
```

Esercizio 6 Siano `a`, `g`, `p`, `r`, `s` e `v` delle URI di un namespace con prefisso `m`. Scrivere un modello RDF che rappresenta le seguenti informazioni: "Le URI `a`, `g`, `p`, `r`, `s` e `v` rappresentano impiegati i cui nomi sono, rispettivamente, Anna, Guido, Patrizia, Ruggero, Stefania, Valerio; inoltre: Guido lavora con Patrizia e Ruggero; Stefania e Ruggero lavorano con due persone che conoscono Anna; infine, Valerio lavora con un impiegato che crede che Guido lavori con Anna". Usare la URI `foaf:name` per esprimere il predicato "ha nome", la URI `m:lavoraCon` per esprimere il predicato "lavora con", e la URI `m:crede` per esprimere il predicato "crede".