Linguaggi e tecnologie per il Web – appello del 13/1/2017

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) Scrivere un documento HTML contenente una form contenente i seguenti campi:

- cognome e nome (casella di testo editabile lunga 40 caratteri)
- matricola (casella di testo editabile lunga 12 caratteri)
- corso di laurea (da scegliere da un menu che riporta alcuni corsi di laurea, incluso IngInfAut)
- tipo corso (selezionabile tramite un menu a due opzioni, T e M)
- email (casella di testo editabile lunga 30 caratteri)
- anno di corso (casella di testo editabile lunga 2 caratteri)
- richieste particolari (area di testo editabile di 12 righe per 60 colonne)
- bottone di invio
- bottone di reset

e in cui vengano effettuati tramite funzioni JavaScript i seguenti controlli:

(a1) verifica che il cognome e nome contenga almeno 5 caratteri. Questa verifica va fatta ogni volta che l'utente cambia il valore del campo cognome e nome;

(a2) verifica che sia stata selezionata una regione. Questa verifica va fatta all'atto dell'invio della form;

(a3) verifica che, se il tipo corso di laurea selezionato è T, allora l'anno di corso sia un numero compreso tra 1 e 3 oppure sia la stringa "FC", e se il tipo corso di laurea selezionato è M, allora l'anno di corso sia un numero compreso tra 1 e 2 oppure sia la stringa "FC". Questa verifica va fatta ogni volta che l'utente cambia o il valore del campo anno di corso o il valore del campo tipo corso;

(a4) verifica che, se il corso di laurea selezionato è IngInfAut, allora o la matricola o l'email sono non vuoti. Questa verifica va fatta ogni volta che cambia il corso di laurea selezionato.

(b) Per ognuno dei controlli specificati al punto (a), dire se è realizzabile in HTML5 senza utilizzare codice JavaScript, e in caso positivo, spiegare come.

Esercizio 2 Scrivere un documento HTML contenente una form contenente i seguenti campi:

- codice fiscale (casella di testo editabile lunga 16 caratteri)
- cognome e nome (casella di testo editabile lunga 60 caratteri)
- data di nascita (casella di tipo date)
- sesso (selezionabile tramite due bottoni radio)
- nazione di residenza (menu a tendina contenente le opzioni Italia, Francia, Spagna)
- bottone di invio
- bottone di reset del local storage degli utenti

Aggiungere funzioni JavaScript e un foglio di stile CSS tali che:

1. i dati relativi ad ogni utente siano memorizzati, all'atto del loro invio, nel local storage;
2. nella parte finale del documento HTML siano visualizzati tutti gli utenti memorizzati nel local storage;
3. il bottone di reset del local storage azzeri il local storage degli utenti;
4. i dati relativi agli utenti di nazionalità diverse siano visualizzati, tramite CSS, con colori diversi. Non si possono usare CSS inlined (solo CSS incorporati o esterni).

Esercizio 3 Data la seguente DTD:

```
<!DOCTYPE r [
  <!ELEMENT r ((a|b),(c|d),(e,f)*)>
  <!ELEMENT a (((e*)|(f*)),d,d+,b)>
  <!ELEMENT b (((b,d,g)|(c,e,f)),g)>
  <!ELEMENT c ANY>
  <!ELEMENT d (#PCDATA|d|r)*>
  <!ELEMENT e #PCDATA>
  <!ELEMENT f EMPTY>
  <!ELEMENT g (#PCDATA|a|b)*>
  <!ATTLIST b attr CDATA #REQUIRED
              attrb CDATA #REQUIRED>
  <!ATTLIST h attrh CDATA #IMPLIED>
]>
```

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 Scrivere un foglio di stile XSL che, dato un documento XML, restituisce il documento tale che: 1) l'elemento radice di input viene copiato in output, e il suo contenuto viene ricorsivamente trasformato; 2) ogni elemento `<x>` che è figlio dell'elemento radice non viene copiato in output, e il suo contenuto viene ignorato; 3) ogni elemento `<y>` che è figlio dell'elemento radice viene copiato in output, e il suo contenuto viene ignorato; 4) ogni elemento `<z>` che è figlio dell'elemento radice viene trasformato in un elemento `zz`, e viene creato per tale elemento un sottoelemento che ha per nome `rootSibling` e che ha un attributo che ha per nome il nome dell'elemento di input corrente e come valore `sibling`. Inoltre, il contenuto dell'elemento di input viene ricorsivamente trasformato; 5) ogni altro elemento figlio dell'elemento radice viene copiato in output, e il suo contenuto viene ricorsivamente trasformato; 6) ogni elemento che è figlio di un figlio dell'elemento radice viene copiato in output come figlio dell'elemento radice, e il suo contenuto viene ricorsivamente trasformato; 7) ogni altro elemento viene copiato in output come figlio dell'elemento radice, e il contenuto di tale elemento viene ricorsivamente trasformato; 8) per ogni nodo di tipo testo che viene letto, viene generato in output un elemento `nodotesto` che ha come contenuto testuale il testo dell'elemento corrente di input.

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

```
<r>
  testo 1
  <z/>
  <x>
    <d><w>testo 2</w></d>
  </x>
  <y/>
  <w>
    <t>testo 3
      <d><g>testo 4</g></d>
    </t>
  </w>
</r>
```

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

```
<r>
  <nodotesto>testo 1</nodotesto>
  <zz>
    <rootSibling z="sibling"/>
  </zz>
  <y/>
  <w/>
  <t><nodotesto>testo 3</nodotesto></t>
  <d/>
  <g><nodotesto>testo 4</nodotesto></g>
</r>
```

Esercizio 5 Dato il seguente documento HTML:

```
<html>
  <body>
    inizio documento...
    <p>link1.htm</p><p>link2.htm</p><p>link3.htm</p>
    <p>doc1.htm</p><p>doc2.htm</p><p>doc3.htm</p>
    <p>url1.htm</p><p>url2.htm</p><p>url3.htm</p>
    <span>visualizza qui link1, link2 e link3</span><br/>
    <span>visualizza qui doc1, doc2 e doc3</span><br/>
    <span>visualizza qui url1, url2 e url3</span><br/>
    ...fine documento
  </body>
</html>
```

modificare il documento HTML in modo da poter attivare, cliccando sull'elemento `<p>` corrispondente, il caricamento asincrono dei 9 documenti HTML `link1.htm`, `link2.htm`, `link3.htm`, `doc1.htm`, `doc2.htm`, `doc3.htm`, `url1.htm`, `url2.htm`, `url3.htm`: i primi tre documenti vanno visualizzati all'interno del primo elemento `span`, il quarto, il quinto e i sesto documento vanno visualizzati all'interno del secondo elemento `span`, e gli ultimi tre documenti vanno visualizzati all'interno del terzo elemento `span`.

Esercizio 6 Scrivere un documento HTML che risolve il precedente esercizio 5 utilizzando JQuery.