Tecnologie Web (6 CFU)

C.d.L. in Informatica

Compito del 1 settembre 2011

Nome:

Cognome:

Matricola:

Team:

□ Non è la prima volta che provo questo esame

Ricapitolo: fare qui sotto una croce sul numero di ciascun esercizio risolto. Se lo si consegna su un foglio protocollo, scrivere in maniera chiara il numero dell'esercizio presso la sua soluzione.

Esercizio		**Punti**	**Voto**
1	Domande di base	12	
2	HTML	6	
3	XSLT	4	
4	Javascript	6	
5	Teoria	4	
Totale		**32**	

Domande di base (12 punti)

Rispondere correttamente ad almeno tre delle seguenti domande:

Considerate la sequente frase: "XML e HTML sono due meta-linguaggi di markup". Spiegare se questa affermazione è vera o falsa e perché.

Dato il seguente frammento XML, scrivere un'espressione XPath che restituisca tutti i compiti che sono stati corretti.

```
<compiti>
        <compito data="2011-09-01" corretto="no" />
        <compito data="2011-02-14" corretto="sì" />
        <compito data="2012-01-15" corretto="no" />
        <compito data="2010-04-24" corretto="sì" />
</compiti>
```

Quali sono e come vengono visualizzati gli elementi HTML indicati dal seguente frammento di CSS

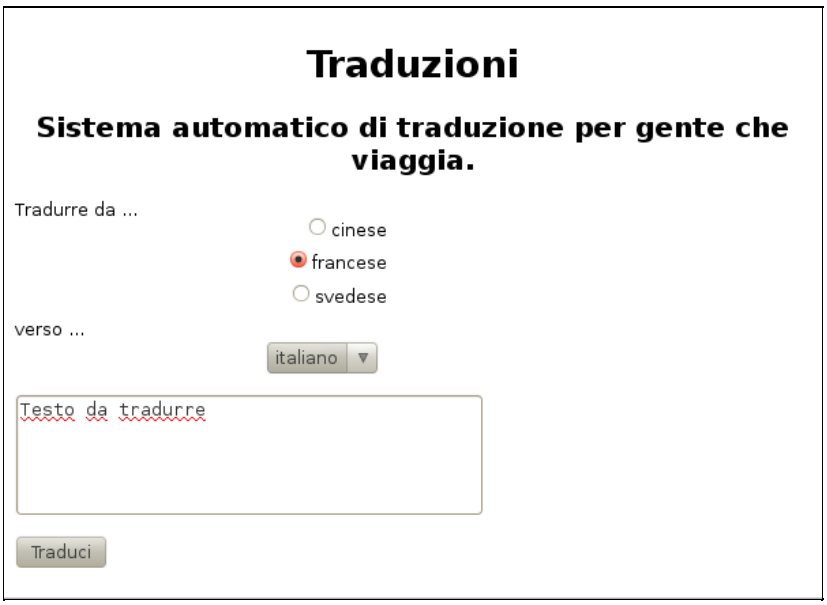
```
body {
    font-size: 12pt;
    text-align: justify;
}

.code {
    font-family:courier, fixed, monospace;
}
```

Che cos'è una tassonomia?

HTML (6 punti)

Scrivere il codice XHTML 1.0 Strict (e CSS) di un documento che rappresenti il contenuto di questa immagine:



Nello scrivere il codice si tengano in considerazione questi vincoli:

- le possibili lingue di destinazione sono "italiano", "tedesco" e "arabo";
- la lingua da tradurre è inviata nella variabile "da", la lingua verso la quale si desidera tradurre nella variabile "verso";
- i valori associati alle varie scelte di lingua possono essere scelti liberamente

Inoltre,

- nessun elemento deve contenere l'attributo `class` (di conseguenza non si possono usare usare selettori classe nel codice CSS);
- nessun elemento deve contenere l'attributo `style`;

Si consiglia di scrivere tutto il codice CSS in un "file" separato, non in elementi `<style>`

Nota: il DocType di XHTML 1.0 Strict è `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">` e il namespace è `http://www.w3.org/1999/xhtml`.

XSLT (4 punti)

Scrivere un foglio di stile XSLT (indipendente dagli identificatori e dai nomi usati per le destinazioni) che trasforma il documento A nel documento B:

Documento A

```
<aereoporto>
    <volo id="AZ12234" ora="10:00">
        <destinazione>Roma</destinazione>
        <prezzo>230</prezzo>
        <posti_totali>100</posti_totali>
        <posti_liberi>10</posti_liberi>
    </volo>
    <volo id="AZ12444" ora="11:00">
        <destinazione>Milano</destinazione>
        <prezzo>200</prezzo>
        <posti_totali>100</posti_totali>
        <posti_liberi>5</posti_liberi>
    </volo>
    <volo id="AZ223" ora="09:00">
        <destinazione>Parigi</destinazione>
        <prezzo>500</prezzo>
        <posti_totali>180</posti_totali>
        <posti_liberi>20</posti_liberi>
    </volo>
    <volo id="AZ12444" ora="09:30">
        <destinazione>Roma</destinazione>
        <prezzo>190</prezzo>
        <posti_totali>100</posti_totali>
        <posti_liberi>2</posti_liberi>
    </volo>
    <volo id="AZ777" ora="17:30">
        <destinazione>Parigi</destinazione>
        <prezzo>340</prezzo>
        <posti_totali>140</posti_totali>
        <posti_liberi>23</posti_liberi>
    </volo>
    <volo id="AZ7655" ora="11:00">
        <destinazione>Roma</destinazione>
        <prezzo>50</prezzo>
        <posti_totali>100</posti_totali>
        <posti_liberi>32</posti_liberi>
    </volo>
</aereoporto>
```

Documento B

```
<destinazioni>
    <destinazione nome="Roma">
        <volo>
            <numero>AZ12234</numero>
            <ora>10:00</ora>
            <posti_venduti>90</posti_venduti>
        </volo>
        <volo>
            <numero>AZ12444</numero>
            <ora>09:30</ora>
            <posti_venduti>98</posti_venduti>
        </volo>
        <volo>
            <numero>AZ7655</numero>
            <ora>11:00</ora>
            <posti_venduti>68</posti_venduti>
        </volo>
    </destinazione>
    <destinazione nome="Milano">
        <volo>
            <numero>AZ12444</numero>
            <ora>11:00</ora>
            <posti_venduti>95</posti_venduti>
        </volo>
    </destinazione>
    <destinazione nome="Parigi">
        <volo>
            <numero>AZ223</numero>
            <ora>09:00</ora>
            <posti_venduti>160</posti_venduti>
        </volo>
        <volo>
            <numero>AZ777</numero>
            <ora>17:30</ora>
            <posti_venduti>117</posti_venduti>
        </volo>
    </destinazione>
</destinazioni>
```

Javascript (6 punti)

Un'API associata ad un sito di social network fornisce un servizio di interrogazione come segue:

- Esempio di richiesta: http://www.socialnetwork.com/find?fieldA=value1,value2&fieldB=value3

  contenente un numero variabile di nomi di campi su cui è possibile eseguire una ricerca (queryable fields) e un numero variabile di valori per ciascun campo (ricercati in OR).

- Esempio di risposta di successo:

```
{
        query: 'segno=Ariete,Pesci',
        success: 'true',
        records: 3,
        seconds: 0.23456,
        queryable_fields: ["id", "nome", "cognome", "eta", "segno", "mail"],
        data: [{
                id: "mr11",
                nome: "Mario",
                cognome: "Rossi",
                eta: "21",
                citta: "Bologna",
                segno: "Ariete",
                mail: "mario_rossi24@gmail.com",
                descrizione: "<p>La stringa di un frammento HTML
                                        <b>arbitrariamente grande</b></p>",
                amici: ["kd27", "qt65", "jh12"],
                info: 'http://www.socialnetwork.com/description?id=mr11'
        },{
                id: "fv21",
                nome: "Filippo",
                cognome: "Verdi",
                eta: "17",
                citta: "Modena",
                segno: "Pesci",
                mail: "Filippo_Verdi@gmail.com",
                descrizione: "<p>La stringa di un frammento HTML
                                        <b>arbitrariamente grande</b></p>",
                amici: [],
                info: 'http://www.socialnetwork.com/description?id=fv21'
        },
                id: "ab78",
                nome: "Antonella",
                cognome: "Bruni",
                eta: "19",
                citta: "Rimini",
                segno: "Ariete",
                mail: "ab_elegant@gmail.com",
                descrizione: "<p>La stringa di un frammento HTML
                                        <b>arbitrariamente grande</b></p>",
                amici: ["ag23", "es41", "wt51","yu76"],
                info: 'http://www.socialnetwork.com/description?id=ab78'
        }]
  }
Esempio di risposta di errore:
  {
        query: 'address!=Via Indipendenza',
        success: 'false',
        records: 0,
        seconds: 0.01176,
        queryable_fields: ["id", "nome", "cognome", "eta", "segno", "mail"],
        reason: "query is malformed or incorrect"
  }
```

La query può essere fatta su uno qualunque dei campi elencati nell'elemento "queryable_fields". Tali campi

possono variare nel tempo. Nella query, ogni campo e ogni valore deve essere correttamente codificato per stare in un URI.

Usando il framework preferito tra JQuery e ExtJs, si scriva il seguente codice javascript:

1. Non appena document.readyState=4 l'elemento deve contenere un form con un elemento di input per ogni campo su cui si può fare query più un pulsante di submit. Attraverso una query erronea (ad es., una query vuota) si può ottenere una lista dei campi utilizzabili. Alternativamente (+1 punto) ci si basi sul local store di HTML 5, memorizzando l'ultima query effettuata e rieseguendola.
2. Non appena una query produce un risultato di successo, l'elemento deve contenere una lista di elementi, uno per ogni record della risposta, preceduti da un titolo che contiene, in linguaggio umano e non informatico o matematico, la query effettuata e il numero di record restituiti. Ogni elemento della lista visualizza nome e cognome della persona, ed è cliccabile. Se la query produce un errore, non deve cambiare la visualizzazione presente al momento.
3. Non appena un elemento della lista del punto precedente viene cliccato, l'elemento   deve contenere una visualizzazione di tutti i campi del record corrispondente. Il campo descrizione contiene HTML ben formato e va visualizzato così com'è. Le informazioni poste all'indirizzo specificato nel campo info debbono essere contestualmente caricate e visualizzate immediatamente (anch'esse sono poste in un documento HTML ben formato). Un link la cui ancora è "Mostra gli amici" punta ad una query che visualizza nella zona delle liste di cui al punto 2 l'elenco degli amici della persona in questione. Il link esiste solo se la lista degli amici è non vuota.

Gli elementi generati dinamicamente debbono essere caratterizzabili tipograficamente in maniera appropriata usando regole CSS, individuali per elementi che compaiono una sola volta, o condivise per elementi che compaiono più volte nel documento.

Teoria (4 punti)

Spiegare le principali differenze tra il markup procedurale e il markup descrittivo.