Final Examination, Semester 1, 2010
600-152 Informatics 2: People, Data, and the Web

Reading Time: 15 minutes. Writing Time: 2 hours.

This paper has 15 pages including this cover page.

Instructions to Invigilators:

- Students will write their answers in one or more script booklets.
- Students may not remove any part of the examination paper from the examination room.

Instructions to Students:

There are 9 questions in the paper. There are a total of 100 marks, making up 50% of your total assessment for 600-152.

- Students should attempt to answer ALL questions.
- Answers must be written in script booklet(s).
- The number of the question should be written at the top of each page of the script booklet.
- The reverse side of any page may be used for notes or draft answers.
- Your writing should be clear; illegible answers will not be marked.
- Your answers can use any Python libraries covered in the subject.

Authorised Materials:

- No materials are authorised.
- Reference materials (e.g. texts, dictionaries) are not permitted.
- Electronic devices (e.g. calculators) are not permitted.

Library: This paper may NOT be held by the Baillieu Library.

Student Number
Question 1: XML and DTD

a) An XML document can be well-formed but not necessarily valid (against a particular schema, or DTD). Describe one situation where validation against a schema is highly desirable, and one situation where it is unnecessary.

[4 marks]

b) The separation of semantic and stylistic elements has benefits for those creating content and those reading it. Discuss the benefits of separating the content from the style for (i) content-creators and (ii) content-consumers.

[6 marks]
Question 2: XML Design

In the Informatics University, the Student Administration Office is using XML to share student records to all departments in the University. The following example shows the kind of student information available through the shared XML data:

<table>
<thead>
<tr>
<th>Student Number</th>
<th>007279</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Name</td>
<td>Ivo Sondergaard</td>
</tr>
<tr>
<td>Emails</td>
<td><a href="mailto:ivos@ugrad.unimelb.edu.au">ivos@ugrad.unimelb.edu.au</a>, <a href="mailto:ivo.sondergaard@googlewave.com">ivo.sondergaard@googlewave.com</a></td>
</tr>
<tr>
<td>Phones</td>
<td>+61 3 8344 1582</td>
</tr>
<tr>
<td></td>
<td>0415 007 279 (mobile)</td>
</tr>
<tr>
<td>Course</td>
<td>Bachelor of Science</td>
</tr>
<tr>
<td>Subjects</td>
<td>Semester 2, 2009:</td>
</tr>
<tr>
<td></td>
<td>600-151, Informatics-1, marks:100</td>
</tr>
<tr>
<td></td>
<td>757-116, Singing-1, mark:40</td>
</tr>
<tr>
<td></td>
<td>Semester 1, 2010:</td>
</tr>
<tr>
<td></td>
<td>600-152, Informatics-2</td>
</tr>
</tbody>
</table>

Design an XML format for representing the student records. Your format should be able to preserve all the information found in the example above and should adhere to the following requirements:

1. The student number, name, and course should always exist.
2. The number of emails should be one or more.
3. There must be at least one phone number or one mobile number.
4. You should decide on the granularity or the detail of your XML design so that the data can easily processed to serve some particular queries such as:
   - getting the list of subjects taken in a particular year;
   - getting the subject code, subject name, and mark from the information on each subject.

You are free to choose the tag names of the elements in your XML format. You are also free to decide which components should be element text and which components should be attributes, but you should follow some of the XML design best practices you have learned in the subject. If in doubt, make and state your assumptions.

You will demonstrate this XML design by:

- writing an XML instance that captures all of the information presented above;
- writing a DTD for your XML design (to help you, some DTD References are provided in the next page).
**Simplified DTD References:**

**DTD Example:**

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE song[
  <!ELEMENT song (title, performer)> 
  <!ELEMENT performer (vocal*, instrument+)>

  <!ELEMENT title (#PCDATA)> 
  <!ELEMENT vocal (#PCDATA)> 
  <!ELEMENT instrument (#PCDATA)>
  <!ATTLIST song id CDATA #REQUIRED>
]> <song id="bigjetplane">
  <title>Big Jet Plane</title>
  <performer>
    <vocal>Angus Stone</vocal>
    <vocal>Julia Stone</vocal>
    <instrument>Angus Stone</instrument>
    <instrument>Julia Stone</instrument>
  </performer>
</song>
```

**DTD Syntax:**

**Elements with children (sequences):**

```
<!ELEMENT element-name (child1, child2, ...)> 
```

**Declaring only one occurrence of an element:**

```
<!ELEMENT element-name (child-name)> 
```

**Declaring at least one occurrence of an element:**

```
<!ELEMENT element-name (child-name+)>
```

**Declaring zero or more occurrences of an element:**

```
<!ELEMENT element-name (child-name*)>
```

**Attribute declaration:**

```
<!ATTLIST element-name attribute-name attribute-type default-value>
```

The attribute-type can be CDATA (the value is character data) or (en1|en2|..) (one from an enumerated list). The default-value can be one of the following: a value, #REQUIRED (the attribute is required), or #IMPLIED (the attribute is optional).
Question 3: Binary Trees

The diagram below shows a binary tree (not necessarily a BST) which holds data about Victorian postal areas and their postcodes.

Assume that we use the postal area (a string) as key. The corresponding value is the area’s postcode. A Python programmer has written the following function to search for a given area’s postcode:

```python
# in tree t, find the value corresponding to key
def search(t, key):
    while not is_empty(t):
        if key == key_of(t):
            return value_of(t)
        elif key < key_of(t):
            t = left_child_of(t)
        else:
            t = right_child_of(t)
    return None
```

It makes use of some auxiliary functions that depend on the exact representation of binary trees. Given a tree t, the function `is_empty` decides whether t is empty or not, `key_of` and `value_of` give the key and the value, respectively of t’s root, and `left_child_of` and `right_child_of` return the left and right sub-trees.

For one or more of the ten areas in the example tree, the function `search` will not behave as expected.

a) Assuming `root` is the root of the binary tree, and that `key` is one of the ten areas used in the example, show a call of the form `search(t, key)` which fails to extract a correct postcode. 

[6 marks]

b) It is possible to transform the example tree to a binary search tree (BST) by swapping two nodes but maintaining the exact same shape of the tree. Which two nodes need to be swapped?

[4 marks]
Question 4: Graphs and Social Networking

a) In the field of social network analysis, **betweenness centrality** and **eigenvector centrality** are two of the numerous measures of centrality of a vertex in a graph (or a network). Describe these two notions of centrality. What can you say about the role in the network of vertices with relatively higher betweenness centrality? Explain in approximately 50-100 words.

b) The **closeness centrality** of a vertex is based on the average of the lengths of the geodesic paths (the shortest paths between two vertices) from that vertex to every other vertex. A lower average for a particular vertex gives it a better closeness centrality.

In the social network depicted below, which vertex has the least central position in the network based on its closeness centrality? In other words, which node has the largest average geodesic distance? Explain your answer in approximately 50-100 words.
Question 5: HTTP and CGI [10 marks]

a) Assume the following HTTP header is part of a message sent from one computer (A) to another computer (B).

HTTP/1.1 200 OK
Date: 21 Jun 2010 01:27:00 GMT
Server: Apache/2.2.13 (Unix) mod_ssl/2.2.13 OpenSSL/0.9.71 DAV/2
Set-Cookie: user=6303c542-7766-4e83-ac3f-666cb47bc37a; Max-Age=604800;
Connection: close
Content-Type: text/plain

Assuming the conventional client-server model, which of A and B would you think is the client? Justify your answer briefly. [4 marks]

b) Below is the source of an XHTML document which provides a form.

```html
<html>
<head>
<title>Sign-up form</title>
</head>
<body>
<h1>Sign-up form</h1>
<form method="get" action="/register.py">
  I would like to help with:
  <br />
  <input type="checkbox" name="catering"> catering
  <br />
  <input type="checkbox" name="cleaning"> cleaning
  <br />
  My name is:
  <input type="text" name="name">
  <br />
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

The Python script register.py is meant to process the data that are submitted via this form.

Outline how register.py may access the submitted data. That is, show, using a few lines of Python code, how register.py can decide which checkboxes were checked and which name was typed in the form. You may either assume that Python’s cgi library is available, or, alternatively, that environment variables can be accessed through the dictionary os.environ. [6 marks]
Question 6: Unicode and Internationalization

a) Describe at least three issues in the localization of software. Explain in approximately 50-100 words.

b) The Unicode character system supports multiple forms of encoding, such as UTF-8 and UTF-16. Both UTF-8 and UTF-16 are variable length encoding. UTF-8 encoding uses one to four bytes to represent a character in the Unicode standard, while UTF-16 uses one or two 16-bit units. In UTF-8, singlebyte is reserved for encoding one of the 128 ASCII characters.

Consider a company that plans to store some corporate documents using XML and Unicode standard. The documents is written in English language. Which encoding should you recommend to be used in the XML files so that the least amount of storage is required? Explain in approximately 50-100 words.
Question 7: Searching and Indexing

In Phase 2 of the Informatics 2 Project, you were asked to store information about movies, by constructing a dictionary for each movie, and “pickling” each dictionary into a file.

For example, here is the dictionary representation of one movie:

```python

{'title': 'Harry Potter and the Goblet of Fire',
 'rating': 'PG',
 'studio': 'Warner Bros.',
 'genre': 'Adventure',
 'cast': ['Daniel Radcliffe', 'Emma Watson', 'Rupert Grint'],
 'writer': 'Steven Kloves',
 'director': 'Mike Newell',
 'date': '2006',
 'origin': 'UK / USA',
 'reviews': [('Washington Post', '80'),
 ('Chicago Sun-Times', '88'),
 ('The New York Times', '80'),
 ('LA Weekly', '60'),
 ('Los Angeles Times', '90'),
 ('Rolling Stone', '75'),
 ('Wall Street Journal', '90'),
 ('Entertainment Weekly', '67'),
 ('Empire', '60'),
 ('Variety', '90'),
 ('Salon.com', '90'),
 ('The Onion (A.V. Club)', '80'),
 ('TV Guide', '75'),
 ('Slate', '100')],
 'earnings': '892194397'
}
```

We assume it is pickled in a file called `movie0001.pickle`.

Suppose a function `safe_load_pickle(movie_pickle)` is available for you. It takes a pickled dictionary file and returns the dictionary of the corresponding movie. Here is an example of how this function behaves.

```python
>>> movie_dict = safe_load_pickle('movie0001.pickle')
>>> movie_dict['title']
'Harry Potter and the Goblet of Fire'
>>> movie_dict['cast']
['Daniel Radcliffe', 'Emma Watson', 'Rupert Grint']
```
Also suppose you have the following definition of a function which will return all file names from a given directory (folder):

```python
import os
def listing(folder):
    path, dirs, filenames = os.walk(folder).next()
    return filenames
```

You can use this to retrieve the names of all files in a folder `data_dir` like so:

```python
for filename in listing(data_dir):
    # do something with the individual filename
```

You can use `safe_load_pickle` and `listing` to answer the following question.

Implement, in Python, a function `ratings` which goes through pickled movie files in a given directory and finds, for each reviewer, the average score given by that reviewer. For example, if the directory 'data' contains a collection of pickled movie files, then `ratings('data')` should return a dictionary such as this:

```python
{
    'The Onion (A.V. Club)': 56,
    'Los Angeles Times': 63,
    'Variety': 60,
    'Slate': 60,
    'Wall Street Journal': 59,
    'The New York Times': 59,
    'Salon.com': 63,
    'Chicago Sun-Times': 71,
    'Entertainment Weekly': 63,
    'TV Guide': 59,
    'Washington Post': 53,
    'Rolling Stone': 65,
    'Empire': 62,
    'LA Weekly': 56
}
```

where, for example, 56 is the average score that LA Weekly assigned across all movies in the directory 'data'.
Question 8: XHTML and CSS

Suppose you are tasked to apply CSS in designing web pages that display student records. In your script booklet, write CSS (Cascading Style Sheets) for the following XHTML file.

```html
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-au">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Student Record of Ivo Sondergaard</title>
<link type="text/css" href="styles.css" rel="stylesheet" />
</head>
<body>
<h1 id="pageTitle">Informatics University Student Record</h1>

<div id="personal">
<h2>Personal Information</h2>
<p>Student Number: <span id="stuId">007279</span></p>
<p>Student Name: <span id="stuName">Ivo Sondergaard</span></p>
<p>Gender: <span id="stuGender">Male</span></p>
</div>

<div id="contact">
<h2>Contact Information</h2>
<p>Address: 111 Barry St, Carlton, VIC, 3010, Australia</p>
<p>Email: ivos@ugrad.unimelb.edu.au, ivo.sondergaard@googlewave.com</p>
<p>Phone: 61 3 8344 1582</p>
<p>Mobile: 0415 007 279</p>
</div>

<div id="academic">
<h2>Academic Information</h2>
<p>Course: Bachelor of Science (R01-AA)</p>
<table id="subject_table">
<thead>
<tr>
<td colspan="4">List of Subjects</td>
</tr>
</thead>
<tbody>
<tr class="semester">
<td colspan="4">Semester 2, 2009</td>
</tr>
<tr class="subject">
<td>600-151</td>
<td>Informatics-1</td>
<td>H1</td>
<td>100</td>
</tr>
</tbody>
</table>
</div>

</body>
</html>
```
<tr class="subject">
    <td>757-116</td>
    <td>Singing-1</td>
    <td>F</td>
    <td>40</td>
</tr>
<tr class="semester">
    <td colspan="4">Semester 1, 2010</td>
</tr>
<tr class="subject">
    <td>600-152</td>
    <td>Informatics-2</td>
    <td></td>
    <td></td>
</tr>
<tr class="subject">
    <td>615-201</td>
    <td>Information Visualisation</td>
    <td></td>
    <td></td>
</tr>
</tbody>
</table>
</div>
<div id="footer">
    <p>For any enquiry about this record, please contact student administration via <span class="email">student-admin@unimelb.edu.au</span></p>
</div>
</body>
</html>

Your CSS should stylise this XHTML page such that:

- The first level heading with #pageTitle id has sans-serif font, 150% in size, and bold in weight.
- The second level headings are formatted in green-colored italic sans-serif font and 120% in size.
- The student emails are displayed in monospace font.
- The email of student administration (in the footer) is in bold monospace font.
- The border of all the cells in the table is styled as black-colored 1-pixel solid line.
- The semester headings (e.g. Semester 1, 2010) in the same table should be formatted as white-colored bold text on black background.
- The subject codes in that table should be colored red and placed on gray background.

The result of your CSS should be similar to the following picture:
Informatics University Student Record

Personal Information

Student Number: 007279
Student Name: Ivo Søndergaard
Gender: Male

Contact Information

Address: 111 Barry St, Carlton, VIC 3010, Australia
Email: ivos@ugrad.unimelb.edu.au, ivo.sondergaard@googlewave.com
Phone: 61 3 8344 1582
Mobile: 0415 007 279

Academic Information

Course: Bachelor of Science (B01-AA)

| Semester 2, 2009 | | Semester 1, 2010 |
|-----------------|------------------|
| 600-151 Informatics-1 | H1 100 | 600-152 Informatics-2 |
| 757-116 Service-1 | F 40 | 615-201 Information Visualization |

Note: Obviously, there are various ways in using CSS to achieve the same presentation outcome. There is no further restriction or constraint as to what combination of selectors you should use. As long as your CSS can produce the required styles in the context of the given XHTML code, your answer would be considered correct.

CSS References:

CSS Rules Format:

selector { property: value; ... }

CSS Selectors:

element selector: element_name
ID selector: #element_id
class selector: .element_class

CSS Advanced Selectors:

descendant selector (space in between selectors): selector1 selector2 (property: value)
child selector (>): selector1 > selector2 (property: value)
adjacent sibling (+): selector1 + selector2 (property: value)
first child pseudo class: selector: first-child (property: value)

Relevant CSS Properties:

font-family, font-size, font-weight, font-style, color, background-color
border, border-width, border-style, border-color
Question 9: Tries and Recursion

Note: While the use of recursion is not required in your solutions to this question, recursive function definitions may be simpler and more readable for some of the solutions.

In the lectures and workshop on data compression and Huffman trees we defined a trie as a binary tree which holds data only in its leaves, and in which the position of a data item is determined by the binary representation of the item’s key.

For example, the trie on the right holds characters as data items. If we think of keys as strings of the characters ‘0’ and ‘1’ then G has key "001" because, to find G, starting from the root of the trie, we need to go left, left, and right.

A simple representation for tries that we used involved Python lists. In the lectures and workshop, each trie also had a weight associated with it. Here we won’t need weights, so we represent a non-empty trie as a list \([t_1, t_2, v]\), where, either

1. \(t_1\) and \(t_2\) are both None, in which case the trie is a leaf holding the value \(v\); or
2. \(t_1\) and \(t_2\) are both different from None, in which case the trie is composite, with \(t_1\) as its left sub-trie and \(t_2\) as its right sib-trie, and \(v\) is None.

For example, the trie above is represented as the following list:

\[
\begin{array}{l}
[ \begin{array}{l}
[ [ \text{None, None, "B"}, [ \text{None, None, "D"}, \text{None}], \\
[ \text{None, None, "G"}], \text{None} \\
[ [ \text{None, None, "F"}, [ \text{None, None, "C"}, \text{None}], \\
[ \text{None, None, "E"}], [ \text{None, None, "A"}, \text{None}], \\
\text{None} \\
\end{array} \\
\end{array}
\]
\]

a) Write a Python definition of a function \(\text{traverse}\) so that \(\text{traverse}(t)\) returns the list of characters stored in trie \(t\). The list should give the characters in the order they are met if we read the trie’s leaves from left to right. For example, for the trie shown above, the result should be \([''B','D','G','F','C','E','A',\]. Note that the function should not do any printing, just produce the required list. (Hint: Recursion may be helpful here.)

[7 marks]

b) Write a Python definition of a function \(\text{lookup}\) so that \(\text{lookup}(t,\text{key})\) will return the character stored at the end of the path determined by \(\text{key}\). For example, if \(t\) is the trie depicted on page 14, then \(\text{lookup}(t, '10')\) should return \('E'\). If the string \(\text{key}\) does not correspond to a leaf in the given trie, the function should return \(\text{None}\).

[7 marks]
c) Write a Python definition of a function `dual` which will produce the mirror image of a given trie. For example, given the trie depicted on page 14, `dual` should produce the trie shown below. (Hint: Recursion may be helpful here.)

[4 marks]

```
       A
      /\  
     E   B
    /\   /\  
   C   F   G
  /\     /\  
 D       D
```

d) Suppose we have built a Huffman tree \( t \) to use in the compression of a message \( \text{msg} \). Explain why the trie \( \text{dual}(t) \) from the previous sub-question could be used for compression and un-compression instead of \( t \). Which, if any, of \( t \) and \( \text{dual}(t) \) can be expected to give the best compression?

[2 marks]