COMP10001 Foundations of Computing
Semester 2, 2016

Tutorial Questions: Week 10

1. Fill in the blanks, so that the HTML code on the left will produce the output on the right when rendered by a web browser.

```
<!DOCTYPE     >
<    >
  <    >
    <     >The Title</     >
  </    >
  <    >
    <  >
      <  >
        <  >
          <  >< >bold</ ></  >
          <  >< >underline</ ></  >
          <  >< >italic</ ></  >
        </  >
      </  >
      <  >< ='' >a link</ ></  >
      <  >
      <       ='smiley.gif' alt='smiley'/>
      </  >
      <  >    entities    </  >
    </  >
  </    >
</    >
```

1. ◦ **bold**
   ◦ underline
   ◦ *italic*
2. a link
3. 
4. <entities>

A:

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
  </head>
  <body>
    <ol>
      <li>
        <ul>
          <li><b>bold</b></li>
          <li><u>underline</u></li>
          <li><i>italic</i></li>
        </ul>
      </li>
      <li><a href=''>a link</a></li>
      <li><img src='smiley.gif'/></li>
      <li>&lt;entities&gt;</li>
    </ol>
  </body>
</html>
```

2. (a) Write a Python function `array2table` that accepts a required argument `array`, which contains a

2D array of values in a list-of-lists representation. `array2table` must **return** a string containing a HTML-encoded table suitable for embedding into a HTML document, presenting the data in `array`.

```
>>> x = [['a',1],['b',2]]
>>> array2table(x)
'<table><tr><td>a</td><td>1</td></tr><tr><td>b</td><td>2</td></tr></table>'
```

A:

```
def array2table(array):
    out = "<table>"
    for row in array:
        out += "<tr>"
        for element in row:
            out += "<td>{}</td>".format(element)
        out += "</tr>"
    return(out + "</table>")
```

(b) Extend `array2table` to take two optional arguments `row_headings` and `col_headings`. If `row_headings` is specified, it should be interpreted as a list of strings representing headings for the rows, and similarly for `col_headings`.

A:

```
def array2table(array, row_headings=None, col_headings=None):
    def generate_cell(val, tag="td"):
        return("<{tag}>{value}</{tag}>".format(tag=tag, value=val))

    def generate_row(row, tag="td", first_cell=""):
        row_out = "<tr>{}".format(first_cell)
        for element in row:
            row_out += generate_cell(element, tag)
        return(row_out + "</tr>")

    out = "<table>"
    if col_headings:
        first_cell = ''
        if row_headings:
            assert len(row_headings) == len(array)
            first_cell = generate_cell(" ")
        out += generate_row(col_headings, tag="th", first_cell=first_cell)
    for i in range(len(array)):
        first_cell = ''
        if row_headings:
            first_cell = generate_cell(row_headings[i], tag="th")
        out += generate_row(array[i], first_cell=first_cell)
    return(out + "</table>")
```