# COMP10001 Foundations of Computing
# Semester 2, 2022

**Tutorial Questions: Week 11**

— VERSION: 1478, DATE: OCTOBER 10, 2022 —

## Discussion

1. What is HTML? What is it used for?
2. Revise the HTML tags you've learned about. How do we use them to format our document?
3. What is an HTML entity? Why are they needed?
4. What is the difference between a static and dynamic HTML page?

   **Now try Exercises 1 & 2**
5. What is an algorithm? Why are algorithms a large area of Computer Science?
6. What are the two criteria with which we can judge algorithms?
7. What is the difference between exact and approximate approaches to designing an algorithm? Why might an approximate approach be necessary?
8. Identify the following as belonging to the exact or approximate approaches to algorithms, and discuss how they approach solving problems with some examples:
   Brute-Force (Generate and Test), Heuristic Search, Simulation, Divide and Conquer

   **Now try Exercise 3**

## Exercises

1. Try visiting your favourite website and viewing the HTML source. How many tags can you recognise? Try editing the source to change the appearance of the webpage (this change will only occur locally on your computer). With knowledge of HTML, you can do a lot more on the web!
2. The function `customise_html` takes three arguments, `template_file` (the filename of a html template, see next page for an example), `context` (a dictionary of strings), `outfile` (a string filename for the output). `customise_html` should find variables within the html template denoted by double curly brackets surrounding a variable name, then replace it with the variable's value from the context dictionary. For example, if the template contained the snippet `<b>{{ shape }}</b>`,
   with `context = {"shape": "triangle"}`, then the output file would contain `<b>triangle</b>`.

   For the `template.html` file on the next page, the function should be able to be called like this to generate an output html file which can be opened in your browser:

```
context = {"name": "Dragon",
           "desc": "A legendary creature. Breathes fire!",
           "power": "100",
           "picture_file": "https://upload.wikimedia.org/wikipedia/commons"
                           "/thumb/6/6b/Easy_origami_dragon_for_beginners-"
                           "_how_to_paint_a_dragon_for_beginners.jpg/320px"
                           "-Easy_origami_dragon_for_beginners-_how_to_pai"
                           "nt_a_dragon_for_beginners.jpg"}
customise_html("template.html", context, "out.html")
```

   All the required lines of the function are available below, out of order. Put each line number in the correct order, and introduce appropriate indentation.

```python
out.write("".join(template))
start = idx + TOKEN_LEN
key = "".join(template[start:end]).strip()
if template[idx:idx + TOKEN_LEN] == ["{", "{"]:
with open(template_file, 'r') as file, open(outfile, 'w') as out:
template = list(file.read()); idx = 0
def customise_html(template_file, context, outfile):
idx += 1
template[idx:end + TOKEN_LEN] = escape(context.get(key, ""))
from html import escape
while idx < len(template):
TOKEN_LEN = 2
end = start + template[idx + TOKEN_LEN:].index("}")
```

`template.html`

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Your new card!</title>
    <nav class="navbar bg-primary">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Card deck</a>
      </div>
    </nav>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
     rel="stylesheet" crossorigin="anonymous"
     integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi">
  </head>
  <body>
    <div class="container-fluid">
      <div class="card" style="width: 18rem;">
        <img src="{{ picture_file }}" class="card-img-top" alt="{{ name }}">
        <div class="card-body">
          <h5 class="card-title">{{ name }}</h5>
          <p class="card-text">{{ desc }}</p>
          <a href="#" class="btn btn-primary">Play</a>
        </div>
        <div class="card-footer">
          Magic level: {{ power }}
        </div>
      </div>
    </div>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
   integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3">
  </script>
  </body>
</html>
```

3. Search the following sorted lists for the number 8, using **(a)** Linear search (Brute-Force approach) and **(b)** Binary search (Divide and Conquer approach)

   Think about the best, worst and average case scenarios of these algorithms. For example, can the best case scenario of a Brute-Force algorithm be faster than running the same task with a more clever algorithm?

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (a) | 1 | 2 | 4 | 5 | 8 | 9 | 10 | 12 | 15 | 19 | 21 | 23 | 25 |
| (b) | 8 | 9 | 11 | 15 | 16 | 17 | 22 | 24 | 27 | 28 | 29 | 32 | 33 |
| (c) | 2 | 4 | 5 | 6 | 7 | 9 | 11 | 12 | 13 | 15 | 19 | 22 | 25 |

## Problems

1. Write a Brute-Force algorithm to solve the following problem:

   The length of a ship is an integer. The captain has sons and daughters. His age is greater than the number of his children, but less than 100. How old is the captain, how many children does he have and what is the length of the ship if the product of these numbers is 32118?

2. Implement linear search and binary search in Python. For an extra challenge, write a recursive version of binary search. Each search function should works as follows: `search([1, 3, 4, 6, 8, 9], 4)` should return `2`, (and `None` if the value is not in the list).

3. Write a function which takes a list of lists containing cell values and optionally two lists specifying row and column headings, and formats the cell values into a HTML table, returning the HTML text as a string. `list_to_table([[1, 2, 3], [4, 5, 6]], ['r1', 'r2'], ['c1', 'c2', 'c3'])` should return a string equivalent to the following HTML.

```html
<table>
    <tr>
        <td>&nbsp;</td>
        <th>c1</th><th>c2</th><th>c3</th>
    </tr>
    <tr>
        <th>r1</th><td>1</td><td>2</td><td>3</td>
    </tr>
    <tr>
        <th>r2</th><td>4</td><td>5</td><td>6</td>
    </tr>
</table>
```