# Homework 2

Due Friday, February 8 at 5:00 PM

For too long apes and monkeys have been under the thumb of man. Well the time as come to oppose that thumb, and take ahold of what is rightfully ours - the world!
-Mojo Jojo, *The Powerpuff Girls (2002)*

## Handing In

To hand in a homework, go to the directory where your work is saved and run `cs0160_handin hwX` where X is the number of the homework. Make sure that your written work is saved as a .pdf file, and any Python problems are completed in the same directory or a subdirectory. You can re-handin any work by running the handin script again. We'll only grade your most recent submission. To install stencil Python files for a homework, run `cs0160_install hwX`. Please leave room between questions and in the margins on your pdf so that your grader can leave feedback on your work. **You will lose points if you do not hand in your written work as a .pdf file.**

## Overview

The following applies for all homeworks:

- We will not accept paper handins. You must submit all written work as a PDF. For information on how to make PDFs of your work, see the very handy **PDF guide**. Please make sure to **NOT** include any identifying information, e.g. name or banner info on your handin, as we are anonymizing grading - we appreciate your help with this.
- Please follow proper pseudocode formatting guidelines. See our **pseudocode standards**. If you use LaTeX, use the `newalg` or `algorithmic` packages or our CS16 `pseudo` environment to format your pseudocode. For even more wizardry, check out the Docs section of our website for a LaTeX handout and tips on improving your pseudocode.

# 1 Written Problems

## Problem 2.1

### Argmax

1. **Silly premise:** Blossom, Bubbles, and Buttercup are on a mission to stop Mojo Jojo from wreaking havoc on Townsville. Lucky for them, they have dicovered a funtion $f$ that tells them the number of monsters in a specific location in Townsville. Help them use this function to stop as many monsters at once as possible.

   ---

   Write pseudocode for the function `argmax(L, f)`, where $L$ is an array and $f$ is a function. Your pseudocode for `argmax` should find and return the element in the array $L$ for which $f$ is maximal. If there are two elements that maximize $f$, return the first. You can assume that elements of $L$ are valid inputs for $f$, and that $f$ outputs real numbers. You can assume that $L$ isn't empty or null.

2. Describe the running time (give us a big-O relation) of `argmax` in terms of $n$ (the size of the array $L$) and $R$ (the worst-case running time of $f$ on any element in $L$), and explain how you came to this conclusion.

## Problem 2.2

### Big $\Theta$ Notation

Demonstrate that $f(n) = 10 + 2^{4 \log_2 n} + 5^{6 \log_5 n}$ is $\Theta(n^6)$. Remember that big-$O$ is an upper bound, big-$\Omega$ is a lower bound, and big-$\Theta$ is a *tight* bound (i.e. upper and lower). Consider how you would show that a function is big-$O$ or big-$\Omega$ of another function by looking at what different integer values $n$ can take on. To prove that $f(n)$ is $\Theta(g(n))$ you must prove both that $f(n)$ is $O(g(n))$ and that $f(n)$ is $\Omega(g(n))$. Make sure you prove this using the formal definition of big-$\Theta$, not just an intuitive explanation.

**Hint**: You may want to simplify your function! This fun property of logarithms might be helpful, $a^{b \log_a n} = n^b$.

# 2 Python Problems

In this case, we have checked for valid input for you, but make sure to do so on your own in the future! Remember, empty lists are valid input!

## How To Test and Run Your Code

For this homework and all subsequent python coding assignments, you will be required to hand in a set of test cases for each python problem. You should

use these tests to confirm that your algorithms work as expected, but they will also be graded according to how comprehensive they are. We will grade your tests by running them against broken implementations of the problems. The more errors your tests catch, the better. When writing tests, try to think of every possible edge case and every kind of input that could be passed into your functions. Keep in mind though that writing many tests which are similar will not earn a better score. Quality over quantity!

We have provided three stencil test files in which you should write your tests: `arraysearch_test.py`, `arraylessthan_test.py`, and `maxword_test.py`. These files have a few example tests filled in to show you how to write your own. Define new functions for your tests, naming them descriptively according to what they are testing for. Fill in these functions with `assert` statements. `assert` takes in a conditional and a string. If the conditional is true, it continues. If it is false, your code stops executing, an error is thrown, and the assert statement string is printed to the console. It is fine for your testing functions to contain multiple `assert` statements as long as they are all related (they should logically fall under the same descriptive testing function name). As a rule of thumb, you should write a new function for each different case you are testing for. Make sure to follow the instructions in the stencil and add the name of each function you write to the list in `get_tests()`.

Examples:
`assert max(1, 2) == 2, 'Test 1 failed'` will pass
`assert max(1, 2) == 1, 'Test 2 failed'` will fail, causing your code to terminate and 'Test 2 failed' to be logged

**Running your tests:** To run your code and your tests, you should run the test files rather than the files in which you wrote your code. For example, to run your `arraysearch` code, you should run `arraysearch_test.py` by typing `python arraysearch_test.py` from your `hw2` directory.

## Problem 2.3

### arraysearch

Implement the following method:

- `array_search` takes in an int and an array (a.k.a. python lists) and returns `True` if the int is in the array. (i.e. `array_search(3, [1, 3, 4]) -> True`). You may **not** assume that the array is ordered.
- Although not necessary, you are allowed to use Python indexing and "slices" (as in array[2:4]) if you wish.
- Do not, however, use the python tool `x in(array)`. While this is an awesome thing to know about (and you should definitely look at the doc-

umentation for it and use it in the future), it would make this problem trivial.

## Problem 2.4

### arraylessthan

Implement the following method:

- `array_less_than` takes in two arrays, `p` and `q`, and returns `True` if `p` and `q` have the same size and each element of `p` is less than its corresponding element in `q`,otherwise returns False. You can assume that the arrays will not be empty.
  Examples:
  `array_less_than([0, 2, 2], [1, 3, 4]) -> True`
  `array_less_than([1, 5, 3], [6, 8, 2]) -> False`

## Problem 2.5

### maxword

Implement the following method:

- `max_word` takes in a string, `s`, and returns the number of occurences of the most common word in the string.
- If the input is an empty string or `None`, then raise in InvalidInputException
- Examples:
  `max_word(‘‘hello world!’’ ) -> 1`
  `max_word(‘‘the quick brown fox jumped over the lazy dog’’ ) -> 2`

Notes:

- You do not have to take into account punctuation or capital letters. All words will be separated with spaces.
- Your solution must run in O($n$) time
- In order to break up the string into individual words, you should look into Python string manipulation. In particular, you will need to use the built-in Python function *string.split()*. Look into what this function does!
- Using the built-in Python *dict* datastructure will be very helpful. A *dict* (short for dictionary) is a data structure that allows you to store an object or value in relation to another object or value. We call this “mapping” a key to a value. As an example, you can think of a real life dictionary as

a mapping between words and their definitions. In an upcoming lecture, we will go into further detail on how dictionaries really work. Here is how you instantiate and use a dictionary in Python:

```
my_dictionary {}
my_dictionary[hello] = 1 #hello is the key, 1 is the value
my_dictionary[world] = 2
print my_dictionary[hello] #prints 1
print my_dictionary[world] #prints 2
```

For this problem, think about what you should use for keys and what you should store in the dictionary

- You will also need to loop through your dictionary once you have filled it in order to find the most common word(s). Read **this** documentation to find out how to traverse your dictionary.