# EECS 281, Week 1: Wednesday

*Hello world! Each week, I will try to bring a handout similar to this one in an effort to make it easier for you to follow along. If you'd like, you can take more detailed notes separately, but I will usually post notes (along with slides and code) on my website (`maximal.io`) as well.*

## Asymptotics

| **Name** | Big O | Big Omega | Big Theta |
|---|---|---|---|
| **Notation** | $f(n) \in O(g(n))$ | $f(n) \in \Omega(g(n))$ | $f(n) \in \Theta(g(n))$ |
| **Informal meaning** | Order of growth of $f(n)$ is less than or equal to $g(n)$ | Order of growth of $f(n)$ is greater than or equal to $g(n)$ | Order of growth of $f(n)$ is $g(n)$ |
| **Example family** | $O(N^2)$ | $\Theta(N^2)$ | $\Omega(N^2)$ |
| **Family members** | $\frac{N^2}{2}$, $2N^2+1$, $\log(N)$ | $\frac{N^2}{2}$, $2N^2+1$, $e^N$ | $\frac{N^2}{2}$, $2N^2+1$, $N^2+183N+5$ |

## CAEN

To configure GCC, execute

```
cd
cat >> .bash_profile
module load gcc/5.1.0
```

then press Control-D.

## Valgrind

## Getopt

## Vector

## Deque

# EECS 281, Week 1: Wednesday

## Boyer–Moore majority vote algorithm

Suppose you are given a vector of values, and want to determine whether or not it has a majority element (one that occurs more than half the time). You can use Boyer-Moore Majority Vote Algorithm. It is stated as follows (adapted from Wikipedia):

```
Create an element most and a counter count with count = 0
For each element x of the input sequence:
       If count = 0, then assign most = x and count = 1
       else if most = x, then assign count = count + 1
       else assign count = count - 1
```

This will determine a possible majority element. You must then count how many times most actually occurs. If it occurs more than half the time, you can then report that most is a majority element, otherwise you cannot.

*Implement* `mooresVoting` *below.*

```
/**
 * Changes most to be a potential majority element.
 * Returns true if most is a majority element,
 * otherwise returns false.
 * The .size() of the data vector must be at least 1.
 */
template <typename T>
bool mooresVoting(const vector<T> &data, T &most) {
    size_t count = 0;
    for (size_t i = 0; i < data.size(); i += 1) {
        if (count == 0) {
            most = data[i];
            count += 1;
        } else if (most == data[i]) {
            count += 1;
        } else {
            count -= 1;
        }
    }
    count = 0;
    for (size_t i = 0; i < data.size(); i += 1) {
        if (most == data[i]) {
            count += 1;
        }
    }
    return count > data.size() / 2;
}
```

*When finished, take a picture of your code and email it to* `maximal@umich.edu`.