# Notes on Assignment 2

CSCI 2270: Data Structures
Section: 202
TA: Sanskar Katiyar

## Overview

The official assignment writeup is very well structured and is aimed towards writing modular, organized code. The writeup gives you the following function prototypes and requires you to complete their definition:

| |
|---|
| `void` **`getStopWords`**`(const char *ignoreWordFileName, string ignoreWords[]);` |
| `bool` **`isStopWord`**`(string word, string ignoreWords[]);` |
| `int` **`getTotalNumberNonStopWords`**`(wordItem uniqueWords[], int length);` |
| `void` **`arraySort`**`(wordItem uniqueWords[], int length);` |
| `void` **`printNext10`**`(wordItem uniqueWords[], int N, int totalNumWords);` |

Let's go through each of these functions and address the requirements:

## Functions

### *Main*

Use all the helper functions provided to you in order to:

- Declare and populate `ignoreWords` array.
- Read word-by-word from a file (*Hint in: Starter code*) and populate the `uniqueWords` array.
- Make sure you check if the `wordItem` already exists: if it does then increment the count. Otherwise create a new `wordItem` element in the array for that string and count = 1.
- The core portion is to check whether the array is full. If yes, then implement the array doubling algorithm. (Dynamically allocate a new array of double the size and copy over the elements from the original array).

### *getStopWords*

Read ***exactly*** 50 words (each word on a separate line) from a file whose filename is provided as a command-line argument.

### *isStopWord*

Simple loop to check if the passed word exists in ignoreWords array. Return `true` if found, else return `false`.

### *getTotalNumberNonStopWords*

Simple loop to sum and return the counts of each `wordItem` object.

### *arraySort*

Do not use the built-in `std::sort` function. Instead:

Modify the Bubble sort algorithm provided to you in the recitation writeup (also available in the code repository) to suit the array of structures. You will need to modify the comparator and the swapping mechanism:

```
void bubble_sort(int A[], int n)
{
    int temp;
    for(int k = 0; k < n-1; k++) {
        // Last k elements are already in place
        for(int i = 0; i < n-k-1; i++) {
            if(A[i] > A[i+1]) {
                temp = A[i];
                A[i] = A[i+1];
                A[i + 1] = temp;
            }
        }
    }
}
```

*Remember*: Since arrays are never passed by value, thus any changes that are made inside the function will be reflected outside as well.

### *printNext10*

- You should assume that the `uniqueWords` array passed to this function is already sorted by the count in the descending order.

- Take care of the index that you address in uniqueWords array. How will you iterate over 10 elements ***past*** N elements in the array? What would be the index?

- In order to print 4 numbers past the decimal point, you will need to use a `fixed` format and `setprecision` with cout (You may need to include the `iomanip` header file)

**Code**: `cout << fixed << setprecision(2) << 12.3456789;`
**Output**: `12.34`

## Common Mistakes

1. **Usage of command-line arguments**: This was a major issue in Assignment 1, for a lot of students. Students ended up typecasting bad values resulting in possible `stoi` errors. Please review how command line arguments work and the syntax they follow.

2. **Uninitialized variables**: Another issue was the failure to initialize variables such as *length* and then passing them to the array as index, resulting in a segmentation fault. Use `-Wall` compilation flag to check for uninitialized variables warning (Alternatively, check for instructor Maciej's announcement on Moodle).

3. **Input and return types**: Look at the function prototype to check what (or if) you are required to return as well as the input arguments it takes.

4. **Reading a file**: Use `ifstream` to open files for reading. `fstream` may cause permission issues on Coderunner (and requires additional flags).

5. **Declaring static arrays**: Don't declare `uniqueWords` as a static array, i.e., `wordItem uniqueWords[100]; //incorrect here`