# COMP 322
## Winter Semester 2023

INSTRUCTOR: DR. CHAD ZAMMAR
chad.zammar@mcgill.ca

---

## Assignment 1: Exploring Functions and Files.

## Due date: 10 February 2023, 11:59 PM.

### Before you start:

- Research for similar problems is tolerated. However, your submission should reflect individual work and personal effort.
- Some of the topics may not be covered in class due to our limited time. You are encouraged to find answers online. You can also reach out to the TAs for guidance.
- Please submit your assignment before the due date to avoid penalties or worse risking your assignment being rejected.
- Submit one file called **assignment1_yourIDNumber.cpp (assignment1_ followed by your McGill ID number)** containing all the functions together with their implementations. It will also contain the main() function that runs everything.

Make sure your code is clear and readable. **Readability of your code as well as the quality of your comments will be graded**.

- No submission by email. Submit your work to mycourse.
- If your code does not compile it will not be graded.
- Be happy when working on your assignment, because a happy software developer has more inspiration than a sad one :).

Software developers use multiple tools to help them achieve their tasks in the most efficient way possible. One of these tools is a “diff tool” used to compare files. This is especially useful when you want to know whether a code file that you were working on had been changed by another developer. All versioning control systems (such as Git) have some kind of diff utility embedded in them.

In this first assignment we will be creating our own diff tool with your beloved programming language C++.

For the sake of simplicity we will limit the comparison to only text files (no pictures, no binaries etc.).

## Question 1 [10 points]

In this first step of the assignment we will be starting very simple. Let’s write a function that compares 2 strings and returns true if they were identical, false otherwise. The signature of this function should be:

```
bool word_diff(std::string word1, std::string word2);
```

For example:

```
std::string str1 = "Hello World";

std::string str2 = "hEllO World";

std::string str3 = "World";

std::string str4 = "Hello World";

bool result = word_diff(str1, str2); // False

bool result = word_diff(str1, str3); // False

bool result = word_diff(str1, str4); // True
```

## Question 2 [20 points]

Now let's follow a classical approach to compare the content of 2 files. We should open each file, read their content word by word and compare them until a first mismatch occurs.

Let's implement a function called **classical_file_diff** that takes 2 arguments each of which is a file name and returns a boolean indicating whether the 2 files are identical or not. The signature of this function should be:

```
bool classical_file_diff(std::string file1, std::string file2);
```

This function **must** use the helper function from Question 1 that compares 2 words.

Example:

Create a folder within the same directory where your executable resides and name it "txt_folder". In it create 2 files: file1.txt and file2.txt.

In file1.txt copy/paste the following lines:

```
My dear C++ class.

I hope that you enjoy this assignment.
```

file2.txt copy/paste the following lines:

```
My dear C++ class.

I hope that you like this assignment.
```

The following code should yield **false** in the variable **result**:

```
    std::string file1 = "./txt_folder/file1.txt";

    std::string file2 = "./txt_folder/file2.txt";

    bool result = classical_file_diff(file1, file2); // False
```

Please note that you may need to modify the slashes in the path name according to the expectations of your operating system.

## Question 3 [10 points]

Now let's push it a bit further and try to optimise. Comparing word by word is time consuming. A better strategy would be to take advantage of hashing functions.

A hash function is a function that maps data of arbitrary size to fixed-size value as per wikipedia: https://en.wikipedia.org/wiki/Hash_function.

Implementing a robust hash function is a tricky thing and out of scope for this assignment. To make things easier we'll take advantage of C++ default hash object used by the standard library **std::hash**.

You are invited to research std::hash in order to understand how it works. In its basic form std::hash can take a string as input and returns its hashed value.

Example:

```
std::string mystr = "I love this assignment";
std::size_t h1 = std::hash<std::string>{}(mystr);
std::cout << h1 << std::endl;
```

Don't mind much of the weird syntax required for std::hash, just use it in a similar fashion to this example.

Now let's write a function called hash_it that takes a string as an input parameter and returns the hashed value. The signature of this function must be:

```
std::size_t hash_it (std::string someString);
```

For example:

```
std::string mystr = "I love this assignment";
std::size_t h1 = hash_it (mystr);
std::cout << h1 << std::endl;
```

## Question 4 [20 points]

Let's take advantage of the hashing function that you implemented in the previous question in order to use it in a smart way to check whether 2 files are identical or not without comparing the content of the 2 files word by word.

Write a function called **enhanced_file_diff** that takes 2 filenames as input parameters and returns true if the content of the 2 files were identical, false otherwise. Keep in mind that this time no content comparison should be used in your implementation (no word to word comparison is allowed). The signature of this function should be:

```
bool enhanced_file_diff(std::string file1, std::string file2);
```

Example:

Use the same files that you have created previously and run the following code:

```
    std::string file1 = "./txt_folder/file1.txt";

    std::string file2 = "./txt_folder/file2.txt";

    bool result = enhanced_file_diff(file1, file2); // False
```

Modify the content on file2.txt to match exactly what's in your file1.txt and rerun the code. It should yield true now.

## Question 5 [20 points]

Up until now, we were able to tell whether 2 files were identical or not. For this question we need to provide more information about where the mismatch is happening.

Write a **recursive** function called **list_mismatched_lines** that takes 2 filenames as input arguments and displays to the screen all mismatched lines in those files.

**This function should use hashing techniques and shall not compare strings to detect mismatch.**

The signature of this function should be:

```
void list_mismatched_lines(std::string file1, std::string file2);
```

Example:

Running the following line of code:

```
list_mismatched_lines(file1, file2);
```

Should print to the screen the mismatched lines only, from both files.

The following output should be seen on the screen:

```
file1.txt: I hope that you enjoy this assignment.

file2.txt: I hope that you like this assignment.
```

Please note that if you have multiple mismatching lines, all of them should be printed out to the screen.

## Question 6 [20 points]

Now let's be more precise and write a function that pinpoints the mismatched words only together with the line number where the mismatch occurred. Here also you should use recursive functions and you should use hashing techniques (no one on one comparison of strings from the 2 files).

Your function signature should be:

```
void list_mismatched_words(std::string file1, std::string file2);
```

By calling this function on your same previous files, the output should be:

```
file1.txt: enjoy (line 2)

file2.txt: like (line 2)
```

Please note that if you have multiple mismatching words, all of them should be printed out to the screen.

**Please use the following main() function for testing:**

```
int main()
```

```
{

    // Q1

    std::string str1 = "Hello World";

    std::string str2 = "hEllO World";

    std::string str3 = "World";

    std::string str4 = "Hello World";

    bool result = word_diff(str1, str2); // False

    bool result = word_diff(str1, str3); // False

    bool result = word_diff(str1, str4); // True

    // Q2

    std::string file1 = "./txt_folder/file1.txt";

    std::string file2 = "./txt_folder/file2.txt";

    bool result = classical_file_diff(file1, file2); // False

    // Q3

    std::string mystr = "I love this assignment";

    std::size_t h1 = hash_it (mystr);

    std::cout << h1 << std::endl;

    // Q4

    bool result = enhanced_file_diff(file1, file2); // False

    // Q5

    list_mismatch(file1, file2); // This should print to the screen the
mismatched lines

    // Q6

    list_mismatched_words(file1, file2); // This should print to the screen
the mismatched words



}
```

**Do not modify the main() function and do not implement any logic in it. All your code should be provided within the provided functions. main() function will be used for testing only. You may only add printing statements (cout) for the purpose of adding more clarity by printing out some desired results.**