# CSCI804

## Object and Generic Programming in C++

Autumn 2016

## Assignment 4 (10 marks)

**Due by 11:59pm Sunday 05 June 2016.**

### TASK ONE (5 Marks): Document Retrieval

The field of information retrieval is concerned with finding relevant electronic documents based upon a query. For example, given a group of keywords (the query), a search engine retrieves Web pages (documents) and display them sorted by relevance to the query. This technology requires a way to compare a document with the query to see which is most relevant to the query.

A simple way to make this comparison is to compute the binary cosine coefficient. The coefficient is a value between 0 and 1, where 1 indicates that the query is very similar to the document and 0 indicates that the query has no keywords in common with the document. This approach treats each document as a set of words. For example, given the following sample document:

"`Chocolate ice cream, chocolate milk, and chocolate bars are delicious`"

This document would be parsed into a set of keywords, where case is ignored, punctuation discarded, {`chocolate, ice, cream, milk, and, bars, are, delicious`}. An identical process is performed on the query to turn it into a set of keywords.

Once we have a query **`Q`** represented as a set of words and a document **`D`** represented as a set of words, the similarity (relevance) between **`Q`** and **`D`** is computed by:

$$relevance = \frac{|Q \cap D|}{\sqrt{|Q|}\sqrt{|D|}}$$

where $|Q|$ and $|D|$ represents the number of words in **`Q`** and **`D`** respectively, $|Q \cap D|$ is the number of words appeared in both **`Q`** and **`D`** (intersection of **`Q`** and **`D`**).

Select appropriate STL containers and write a program that takes a set of keywords (any number of words) that represent a query. The program should then compare the query to ***all*** the document files (whose names end with extension **`.txt`**) specified in the file called **`listofdocs.txt`** and output the relevance and the documents in a descending order of the relevance. If a document contains more than 10 words, then just output the first 10 words of the document and a symbol "…" at the end.

For this task you should submit **`DocRetrieval.cpp`**. Your code must compile on Banshee with the instruction

```
$ g++ DocRetrieval.cpp -o DocRetrieval
```

and should run as

**$ ./DocRetrieval** *keywords1 keywords2 keywords3 …*

For example, if the **listofdocs.txt** lists four documents that are in the same directory as the program:

```
Kyle01.txt
Kyle02.txt
Kyle03.txt
Kyle04.txt
```

Note: check the text files for their contents.

Run the program as follow

**$ ./DocRetrieval *kyle radio 2Day girl***

The output would look like:

```
(Kyle04.txt - 32.44%) THE radio network Austereo has pulled the top-rating
2Day FM …

(Kyle03.txt - 23.15%) THE top-rating radio station 2Day FM and its owner,
Austereo …

(Kyle01.txt - 8.98%) The Ten Network has dumped embattled host Kyle
Sandilands as …

(Kyle02.txt - 0.00%) Word around the traps yesterday was that Monday night's
televisual …
```

## TASK TWO (5 marks): Movie Ratings

You have collected files of movie ratings where each movie is rated from 1 (bad) to 5 (excellent). The first line of each file is a number that identifies how many ratings are in the file. Each rating then consists of two lines: the name of the movie followed by the numeric rating from 1 to 5. Here is a sample rating file with four unique movies and seven ratings:

```
File: ratings.txt
----------------------
7
Harry Potter and the Order of the Phoenix
4
Harry Potter and the Order of the Phoenix
5
The Bourne Ultimatum
3
Harry Potter and the Order of the Phoenix
4
The Bourne Ultimatum
4
```

```
Wall-E
4
Glitter
1
------------------------------
```

Choose a proper STL container and write a program that reads multiple files in this format, calculates the average rating for each movie, and outputs the average along with the number of reviews. Here is the desired output for the sample data:

```
Glitter: 1 review, average of 1/5
Harry Potter and the Order of the Phoenix: 3 reviews, average of 4.3/5
The Bourne Ultimatum: 2 reviews, average of 3.5/5
Wall-E: 1 review, average of 4/5
```

For this task you should submit **`Movies.cpp`**. Your code must compile on Banshee with the instruction

**`$ g++ Movies.cpp -o Movies`**

and should run as

**`$ ./Movies ratings1.txt ratingg2.txt ratings3.txt`**

## Submission

Assignments are submitted electronically via the ***submit*** system. For this assignment you must submit all the files via the command:

$ submit -u your_user_name -c CSCI804 -a 4 **`DocRetrieval.cpp Movies.cpp`**

and input your password.

Make sure that you use the correct file names. The Unix system is case sensitive. You must submit all files in one ***submit*** command line.

**Your program code must be in a good programming style, such as good names for variables, methods, classes, and keep indentation.**

**Submission via e-mail is NOT acceptable.**

After submit your assignment successfully, please check your email of confirmation. **You would loss 50%~100% of the marks if your program codes could not be compiled correctly.**

Late submissions do not have to be requested. Late submissions will be allowed for a few days after close of scheduled submission (up to 3 days). Late submissions attract a mark penalty; this penalty may be waived if an appropriate request for special consideration (for medical or similar problem) is made via the university SOLS system *before* the close of the late submission time. No work can be submitted after the late submission time.

A policy regarding late submissions is included in the course outline.

The assignment is an **individual assignment** and it is expected that all its tasks will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during tutorial classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for that assessment task.