School of Computing & Information Technology

CSCI251 Advanced Programming

Assignment 3

Deadline: 23:55pm, October 25, 2024

This assignment is to write a C++ for hexadecimal numbers processing by using generic programming. Given different types of documents (e.g., bonds, certificate, and contracts), the task is to compute some statistics based on similarity between documents and character frequencies. For simplicity, we assume that there is only one type of documents, that is *certificate*. You need to define two classes, namely, *Certificate* and *DocumentHandler.* The first class is for recording information of each certificate and similarity computation whereas the second class is for computing some statistics based on all certificates.

Roadmap. In Section 1 and Section 2, we will describe the Certificate class and the DocumentHandler class respectively. In Section 3, we will describe how to run your program and what your program should output. In Section 4 and Section 5, general notes and submission guidelines will be described.

Section 1: the Certificate class

A certificate consists of many two-digit words. Each word should be a mix of hexadecimal digits from 0 to 9, and A to F (e.g., A8, FC, 24, and 4C). The following methods should also be provided.

- A constructor: this class should have a parametric constructor, while the input argument is ***len***. This constructor will produce random two-digit words, with the total number of ***len***. These words should be a mix of hexadecimal digits from 0 to 9, A to F, and it will be considered as the content of this certificate.
- A container: it should store the words and spaces between words (i.e., all content in this document).
- A destructor: it should output notice of object destruction.
- A display function: this method should display the content of this certificate.
- A similarity function: this method should take another certificate instance to compare. The result is the number of ***related words*** that appear in both certificates. ***Given a word w1 from certificate 1, and a word w2 from certificate 2, w1 and w2 is related if their first digits are the same.***

Example: the similarity between two certificates below is 4, since we have 4 pairs of related words (i.e., CF and C0, FB and FA, 40 and 4A, and 4F and 4A).

```
             8C CF FB 0E AD A7 40 4F A6
(vs)          3F 9E FA DF 63 99 5B 4A C0
```

Section 2: the DocumentHandler class

This class should be used to store and process a collection of documents of the same type. Therefore, it should be a template class and the input documents of the same type should be treated as a generic type T. For simplicity, we only assume there is only one document type (i.e., certificate). However, this class must treat input certificate as a generic type T, and we can also assume that our provided member attributes and functions in these two classes are general and can work with different types of documents.

The following methods and attributes should be provided.

- A container which stores all document objects of the generic type T.
- A display function: this method should display the content of all stored documents.
- A minSimilarity function: this method should determine the minimum number of related words across all pairs of documents in the container.
- A digitStatistics function: this method should determine digits with the minimum and maximum frequency. It is possible that there are multiple digits with the same maximum or minimum frequency. Report all of them.

Example of illustration: we have three lines below and each line refers to one certificate with 6 words.

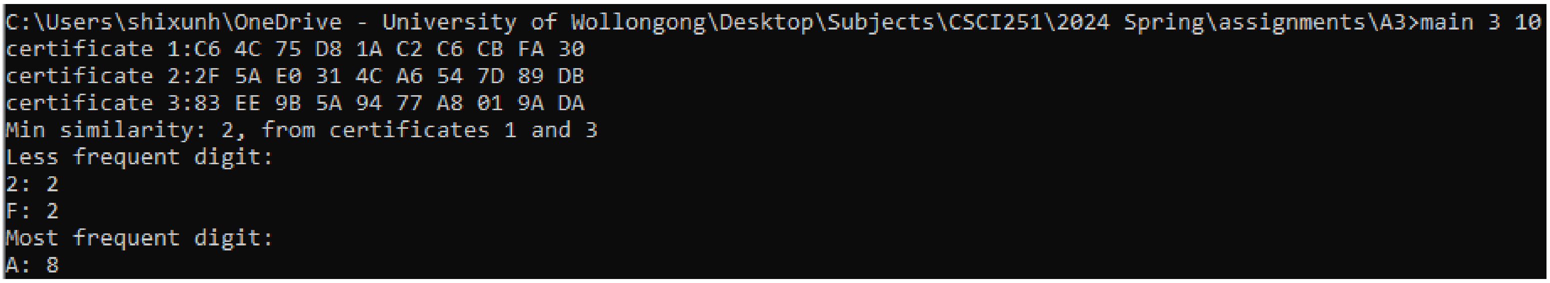
```
            B6 EB FB 4E 35 A6
            3C A7 4C AA 3B 4D
            10 23 44 D1 D2 5E
    For 1st and 2nd certificates, we have:
            B6 EB FB 4E 35 A6
      (vs)  3C A7 4C AA 3B 4D
         Similarity: 6

    for 1st and 3rd certificates, we have:
            B6 EB FB 4E 35 A6
      (vs)  10 23 44 D1 D2 5E
         Similarity: 1

    for 2nd and 3rd certificates, we have:
            3C A7 4C AA 3B 4D
      (vs)  10 23 44 D1 D2 5E
         Similarity: 2
As such, we have minSimilarity = 1;

For digitStatistics, the result will be:
    Less frequent digits with count:
    0 : 1
    7 : 1
    F : 1
    Most frequent digits with count:
    4 : 5
```

Section 3: The Program Execution and Output

Given your executable program ‘main’, the number ***num*** of certificates to be generated and the number ***len*** of words in each certificate, you program should run like below:

./main num len

Program output: you should display all certificates’ content, followed by minimum similarity, the least frequent digits and most frequent digits with frequency statistics. Below is an example output:



```
C:\Users\shixunh\OneDrive - University of Wollongong\Desktop\Subjects\CSCI251\2024 Spring\assignments\A3>main 3 10
certificate 1:C6 4C 75 D8 1A C2 C6 CB FA 30
certificate 2:2F 5A E0 31 4C A6 54 7D 89 DB
certificate 3:83 EE 9B 5A 94 77 A8 01 9A DA
Min similarity: 2, from certificates 1 and 3
Less frequent digit:
2: 2
F: 2
Most frequent digit:
A: 8
```

Section 4: General notes

1. Your assignment should be sensibly organised with the same kind of expectations as previous assignments.
2. Other than the initial command line input, the program should run without user input. This means there should not be pauses during the running time.
3. A character can be represented as an integer. You can search ASCII table online for more details.
4. ***You must use classes and the DocumentHandler class must be a template class and handle certificate objects as a generic type.***
5. ***For all of your self-defined functions, they should be sensibly warped into classes as member functions.***
6. ***You are allowed to use any built-in containers.***
7. ***Make sure that your code is C++17 compliant and can compile on Capa.***

Section 5: Submission Guidelines

Please submit a zip file via Moodle and it should only contain all your source files and a Readme file.

1. Late submissions will be marked with a 25% deduction for each day, including weekends.

2. Submissions more than four days late will not receive marks, unless an extension has been granted.

3. If you need an extension, apply through SOLS before the assignment deadline.

4. Academic misconduct is treated seriously. Specifically, any plagiarised work will be awarded a zero mark and reported to the University.