Module: C++

CA: 3 – Pair Project

Value: 30%

Due Date: See Moodle

Objectives:

- To practice the design and implementation of a C++ application.
- To gain experience with the use C++ Standard Library (SL) container classes (incl. vector, list, hash_set, hash_map).
- To demonstrate the use of advanced C++ concepts (incl. function pointers, smart pointers, and the date and time library).

Functional Requirements:

Each student group is required to develop the code found on a simplified email server for a new company called MyMail. The solution is described as *simplified* because it does not contain actual network connectivity but rather simulates how emails could be created, stored, accessed, and modified on the server-side.

An *user* is defined by the information in the table below:

Variable	Type	Validation	Example
Email	String	Valid email (regex)	"jane.smith@mymail.com"
Password	String	Length >= 8 characters; Must contain minimum of 1 number and 1 letter(non-case senstive).	"Mond4y16"
User Name	String	Length >= 8	"Jane Smith"

An *email* is defined by the information in the table below:

Variable	Type	Validation	Example
Sender	String	Valid email (regex)	"jane.smith@mymail.com"
Recipient(s)	String	Valid email(s) (regex)	"beatrix@mymail.com" or "max@mymail.com; beatrix@mymail.com;..."
DateTime	C++ time classes	Valid time object	25/11/16 at 11:42 PM
Subject	String	(optional)	"Lunch"
Body	String	Length >= 1 character	"Hi, Lunch at 1pm in the Crowne Plaza? Regards, J."
Attachment(s)	Attachment	Properly instanciate Attachment object(s)	Text file, Logo.png

An *attachment* is defined by the information in the table below:

Variable	Type	Validation	Example
FileName	String	Windows valid file name (regex)	"report", "logo"
FileSuffix	String	2-3 character alphanumeric suffix (regex)	"txt", "gif"
FileData	char []	Non null	Character stream from file

The entities above need to be implemented as classes in your solution. Each class must include the following:

- Constructors (full and default) and destructor, with validation, where appropriate.
- Getters & Setters and Print methods
- Overloaded operators (==, !=, =, <<, and >>)
- Overloaded copy and copy assignment operator.

(12 Marks)

Your solution must allow the user to perform the following file operations through a UI:

- **New** - Create a new email object and add any attachments.
- **Send** - Send the email to a single or multiple recipients.
- **View** - Query and view emails for a particular email name.
- **Delete** - Delete an email from those returned for a specified email address.
- **Delete All** - Delete all emails for a specified email address.
- **Search By -** Find and view all emails with a user specified filter. You must implement three filters (e.g. by date range, by subject, by presence of attachment(s)) where one filter uses a regular expression.
- **Reset –** Removes all email data from the system and frees all objects from RAM.

(42 Marks)

In addition, your application must demonstrate **appropriate** usage of the following C++ features and technologies:

- Use of one or more SL container classes (excl. Vector).
- One example of the use of an array of function pointers.
- Use of at least one C++ 11 style casts.
- Use of an algorithm in one of the container classes.
- Use of a template class.
- Use of new and delete for the allocation of objects, with operations on the objects using pointers.
- Use of one of the following C++ 11 SL functions: all_of, any_of, none_of with a suitable predicate.

(16 Marks)

UI Usability Requirements:

The application will require the development of a simple DOS based GUI Please be aware that 10 marks are available for the usability of the final application. The project will be awarded marks based on the usability of the applications. Usability will be measured based on the following criteria:

- **learnability** (i.e. an intuitive user interface requiring a minimum of external support)
- **error tolerance** (i.e. the application recovers gracefully from an input error

Students must consider how best to design the application to maximise usability. Remember there is a clear link between a product's usability and success in a commercial market.

(10 Marks)

Additional Requirements:

An additional 10 marks will be awarded for the efficiency of your implementation which depends on your choice of C++ container class.

Finally, 10 marks will be awaded for the maintainability of your code. Maintainability may be defined by how well you subdivide blocks of functionality into smaller testable functions; your naming convention for classes, methods, variables, and constants; and the level of commenting within your code.

(20 Marks)

Submission Requirements:

1) Source code must be submitted in the relevant sub-folders inside a **single** ZIP file through Moodle.
2) Each student group will be required to attend an interview after the deadline date. Each student will be questioned on the functionality of the code.
3) If an individual fails to perform satisfactorily at interview then a penalty mark will be applied to that student's final grade. Therefore, it is possible for one student in the group to receive full marks while the other student in the group is penalised for a failure to demonstrate an understanding of the submitted work.
4) The assignment **must** be entirely the work of each student group. Student groups are **not** permitted to share any pseudocode or source code from their solution with any other individual or group in the class. Students may **not** distribute the source code of their solution to any student outside of their group in **any** format (i.e. electronic, verbal, or hardcopy transmission).
5) Plagiarised assignments **will** receive a mark of zero. This also applies to the individual/group allowing their work to be plagiarised.
6) Any plagiarism **will** be reported to the Head of Department and a report will be added to your permanent academic record.
7) Late assignments will **only** be accepted if accompanied by the appropriate medical note. This documentation **must** be received within 10 working days of the project deadline. The penalty for late submission is as follows:
    - Marked out of 80% if up to 24 hours late.
    - Marked out of 60% if 24-48 hours late.
    - Marked out of 40% if 48-72 hours late.
    - Marked out of 20% if 72-96 hours late.
    - Marked out of 0%, if over 96 hours late.

8) Each student group **must** complete and sign a single assignment cover sheet. Please submit the signed cover sheet before 5pm on the Friday of the week of the deadline.