**Email Login System**

**Abstract**

Our mission is to develope an object-oriented console application for user to handle their email. This application provide user an environment to build a database of the emails, displaying and handling the database content. However, this program application is not able to send or receive any emails.

**1. Introduction**

Our assignments have mostly finished all the requirements stated, including

* (i) check the email input by the user
  + check only one character ‘@’ is inside an address with some characters before and after it (e.g., abc@def.com)
* (ii) sort the email in ascending order based on the subject inside a database
  + perform the first letter of subject into capital letter, and sort the email with respond to subject in ascending order
* (iii) login the system by checking username and password
  + for new user, create a new account to access the database
  + for old user, login with correct password to acces the database
* (iv) output the database to a file and read the file.
  + save the user input to the file and overwrite the file again
  + read the file and input the file data into database

In our report, we will explain the above details one by one

**2. Methodology**

**i. Divides of work**

* Define the class and its member function
* Develop a static library
* Develop a test mode user interface
* Comment on the program
* Debug the program
* Write report

**ii. Schedule**

We define the program into three stage

1. Design the flow of the project

* 1 Day for preparing own idea
* 1 Day for us to discuss

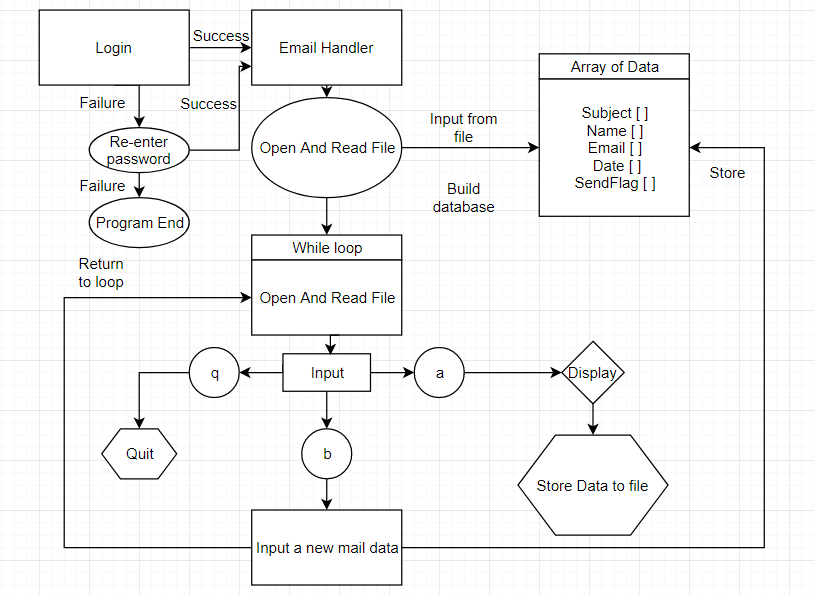
1. Write the function and debug the program

* 2 Day for writing the function
* 2 Day for debugging the program

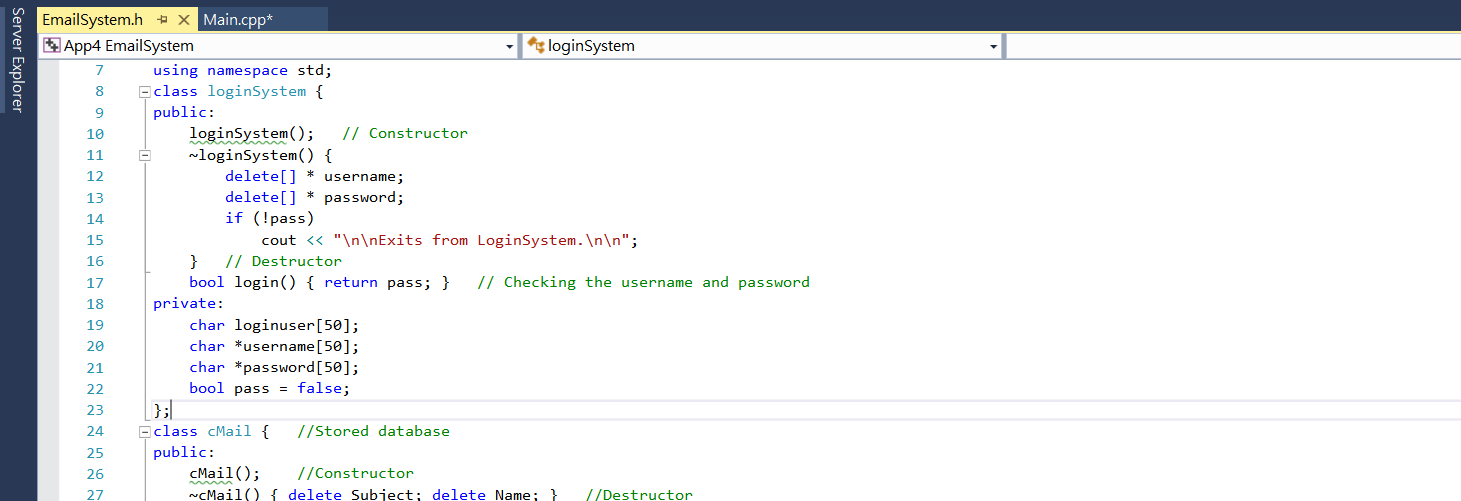
1. Finalise the program into a user-friendly interface

* 1 Day for testing the program
* 1 Day for modify the user-interface

**iii. Structure of the developed program**



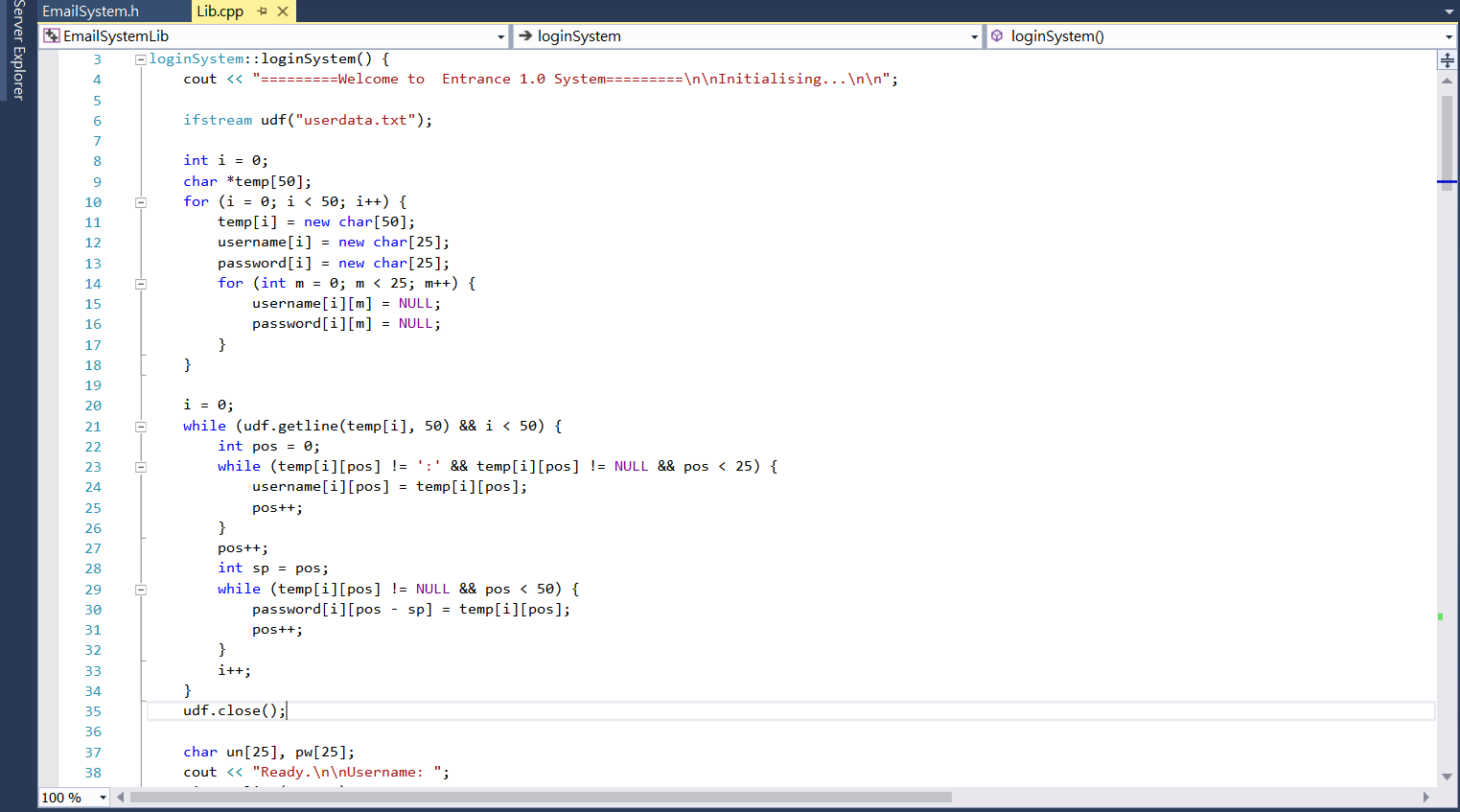
Class:Login System👇

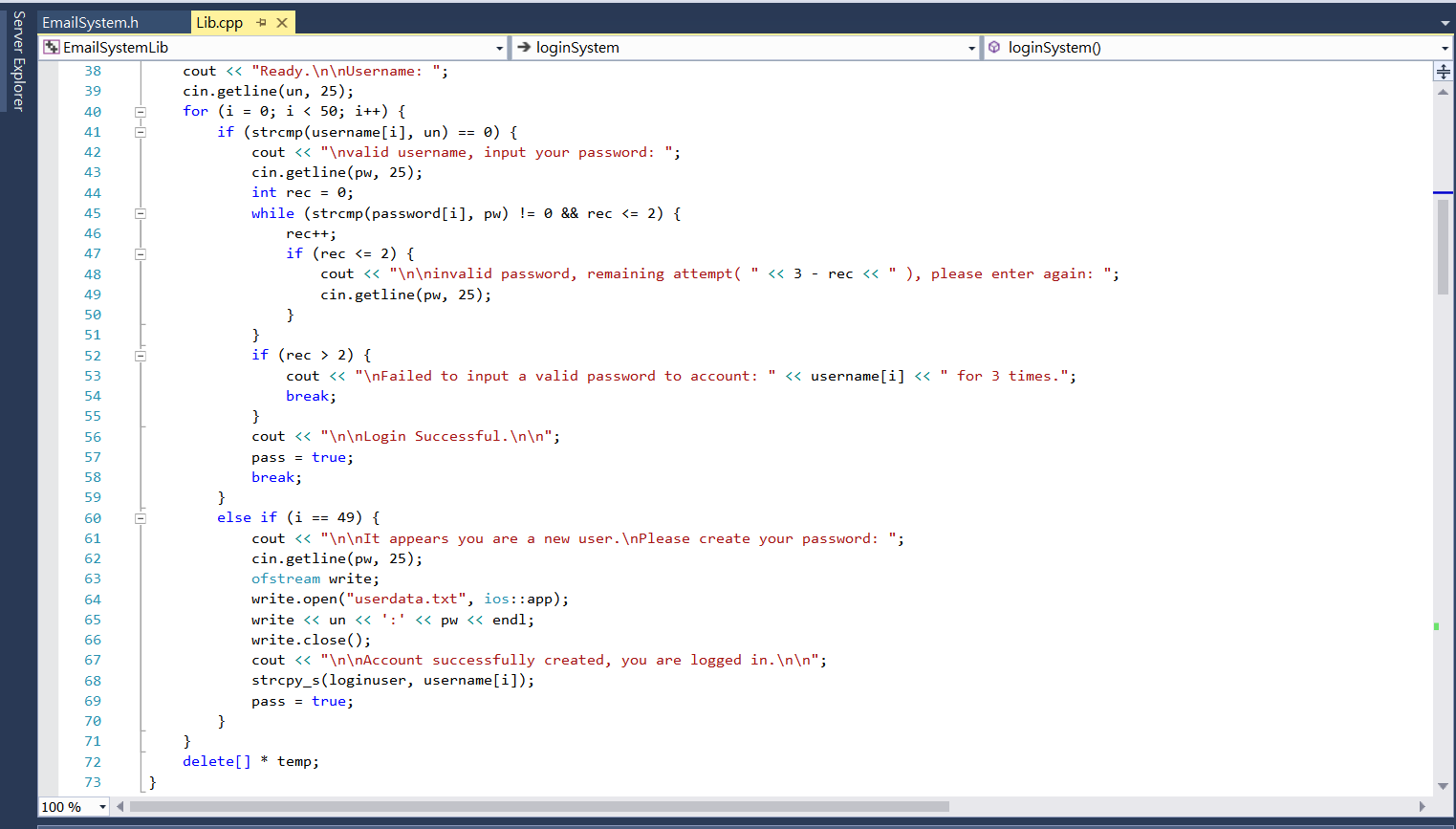


* LoginSystem::login()  
  This function returns a boolean value which indicates if the login information provided by the user is valid.
  + LoginSystem::LoginSystem()
  + This is the constructor of the login system, it also serves as the main code of the system itself.
  + In the code, the system first opens the userdata.txt to retrieve all login informations and form 2 separate arrays, an array of user ID, and an array of password.
  + Then the system will ask the user to input their user ID, then it checks is the inputted data matches the array of user ID. If so, the system will ask for an input of password, in case the password resembles the one associated to the user ID, the return value of login() will be set to true. If the inputted user ID doesn’t match with all elements from the array, the system will move on to create a new account for the new user ID, which ask an input of a new password, and save both value to the userdata.txt.
  + The database userdata.txt stores data in forms of:

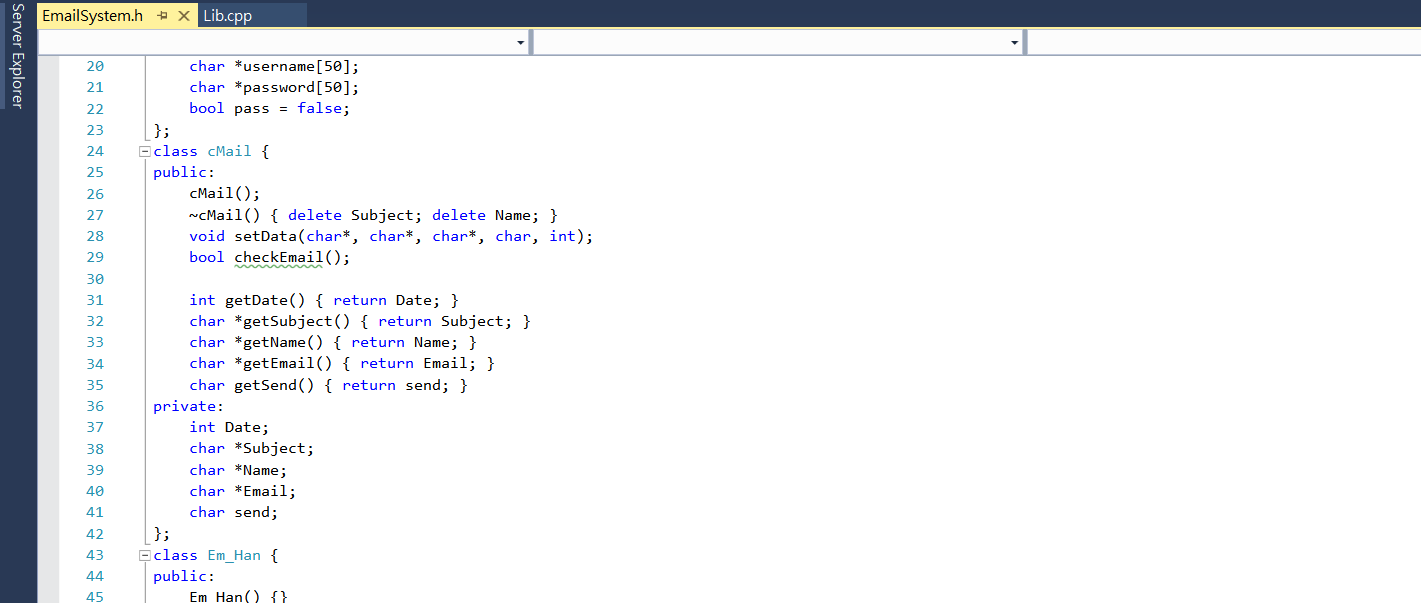
#user ID#:#password#

The system will automatically break the line by “:” to store the first part to the user ID array, and the second part to the password array.





Class :cMail👇



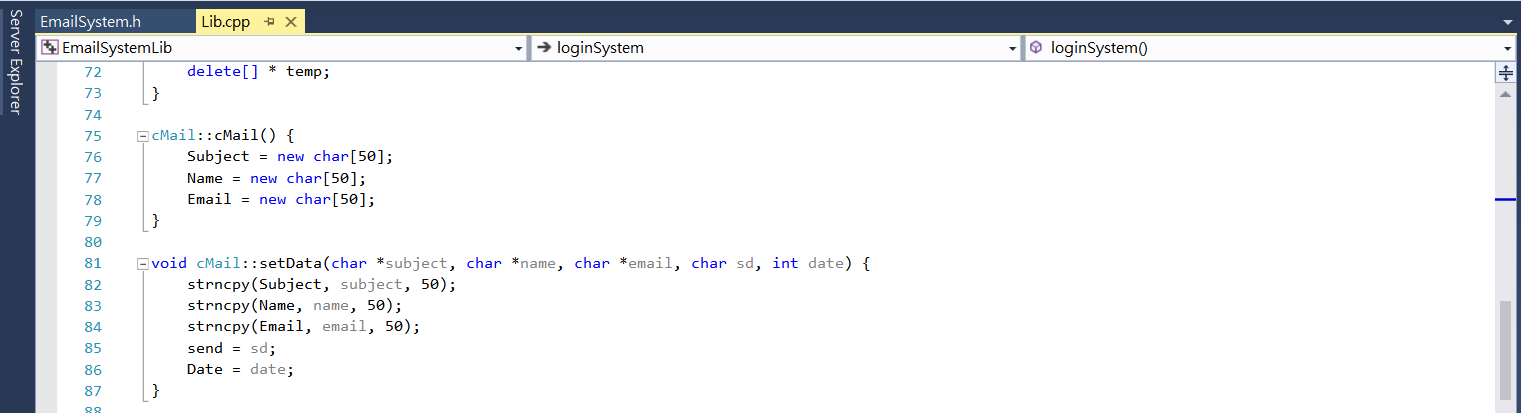
* cMail::setData()

This function will set all variable associated to the object cMail to the parameter provided.

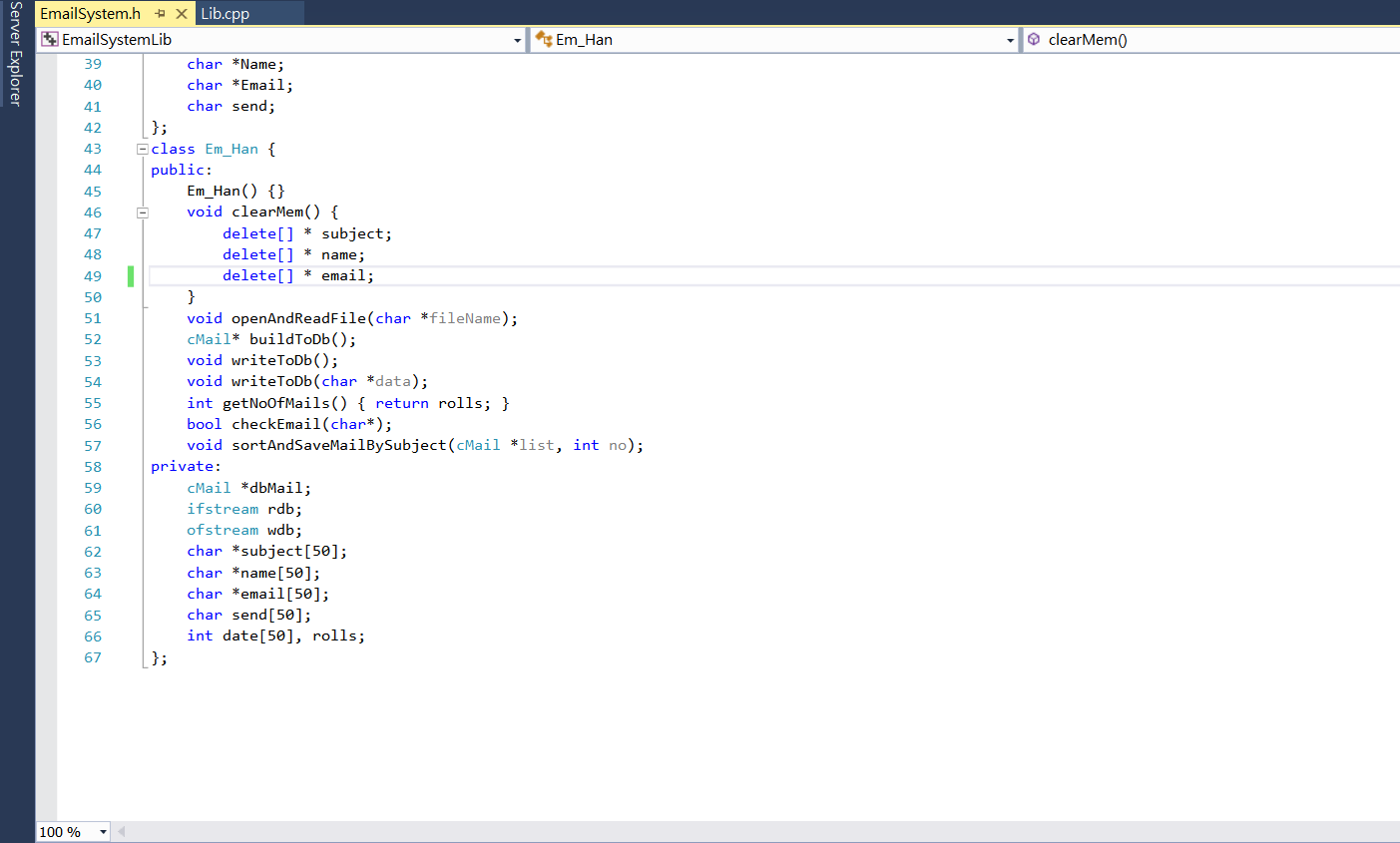
* cMail::checkMail()

This function checks if the email provided in char\* matches the format:

[xxx@xxx.xxx](mailto:xxx@xxx.xxx)



Class :Em\_Han👇



* Em\_Han::openAndReadFile()

This function will open the file pointed by the filename provided by the parameter,

then it uses the same method of checking the database of the login system, with the

5 arrays of data which represents the 5 variables of the cMail object. After it reads the

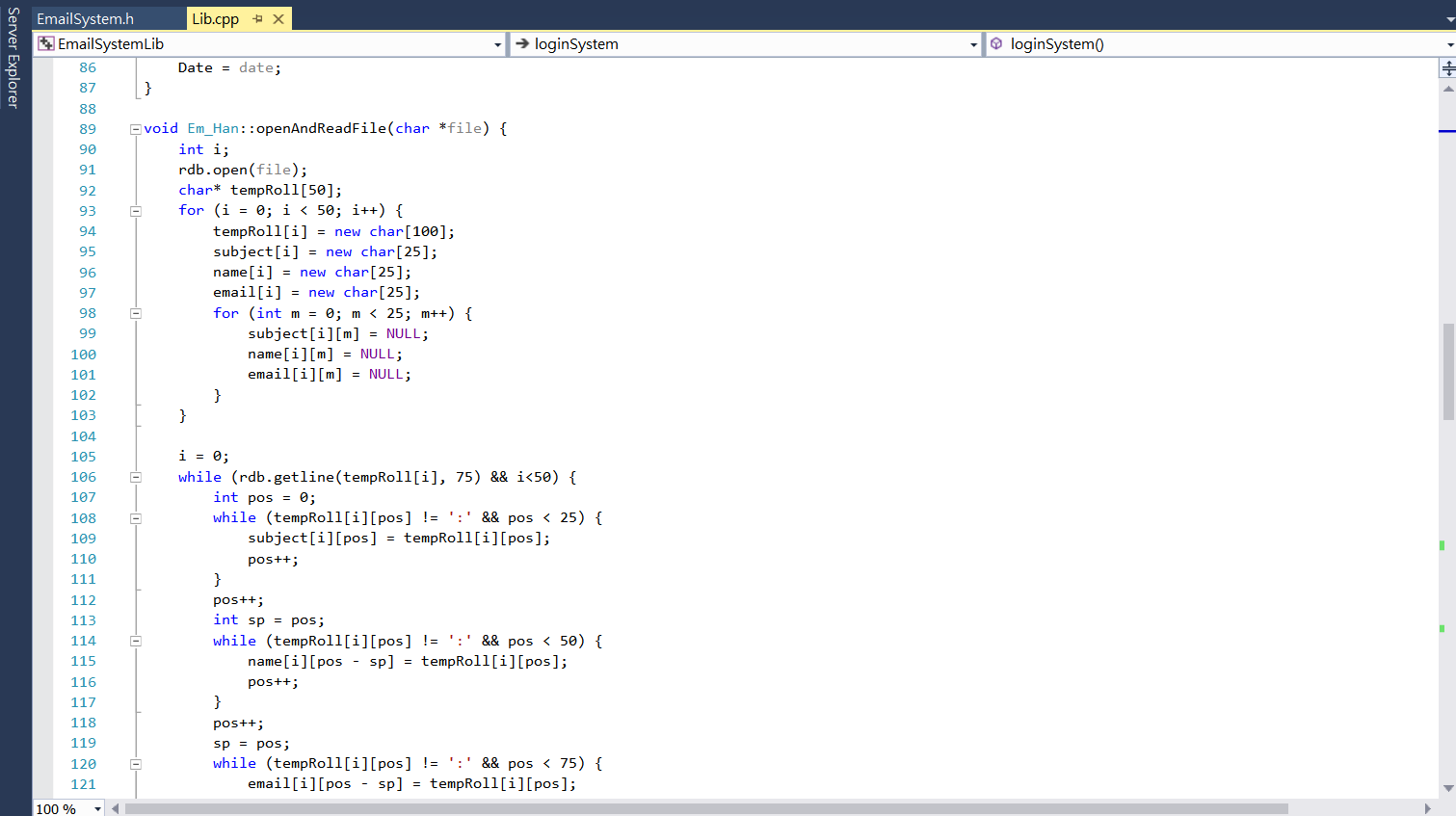
data, it will store them into those array mentioned above.

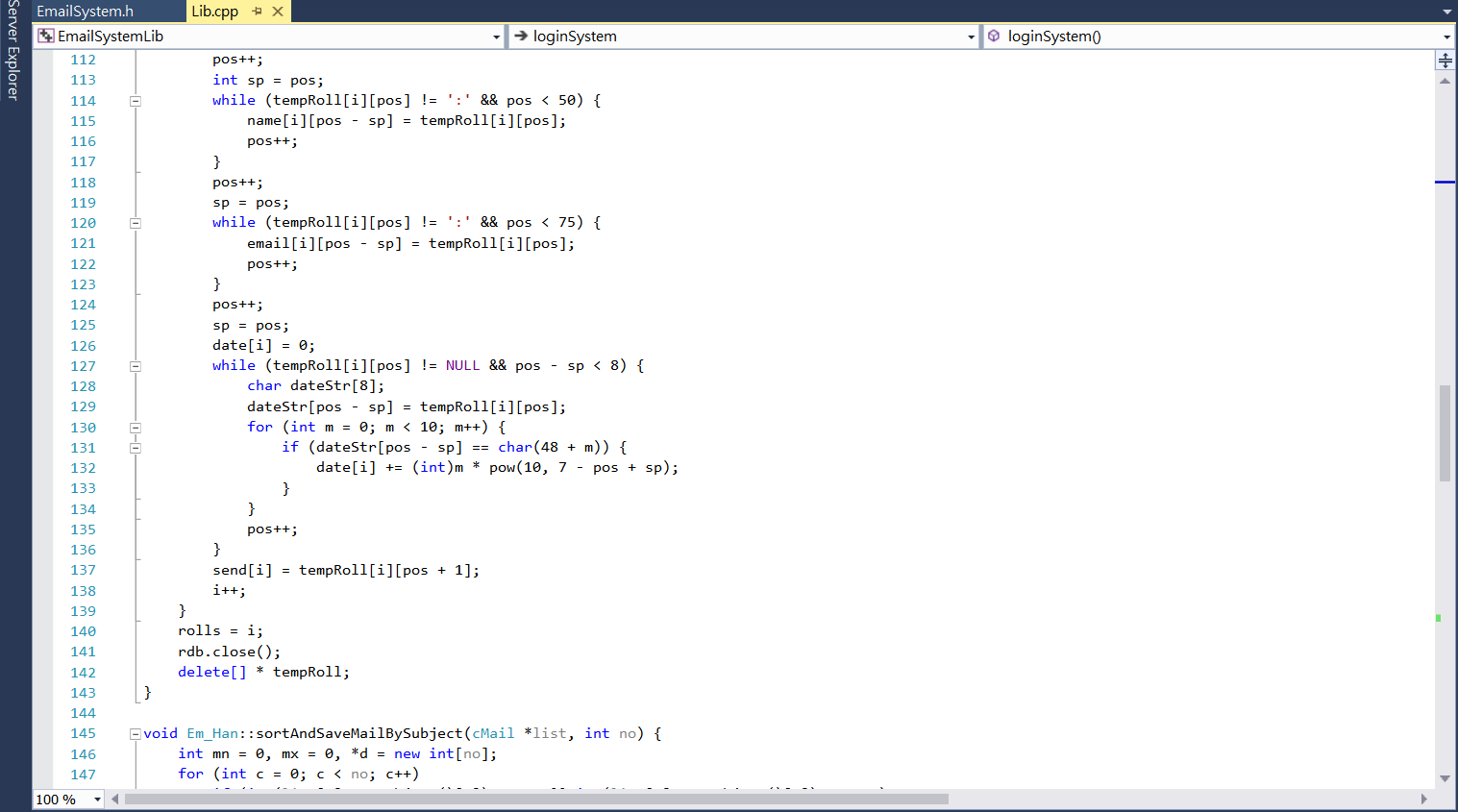
The format of the database used to store the mails:

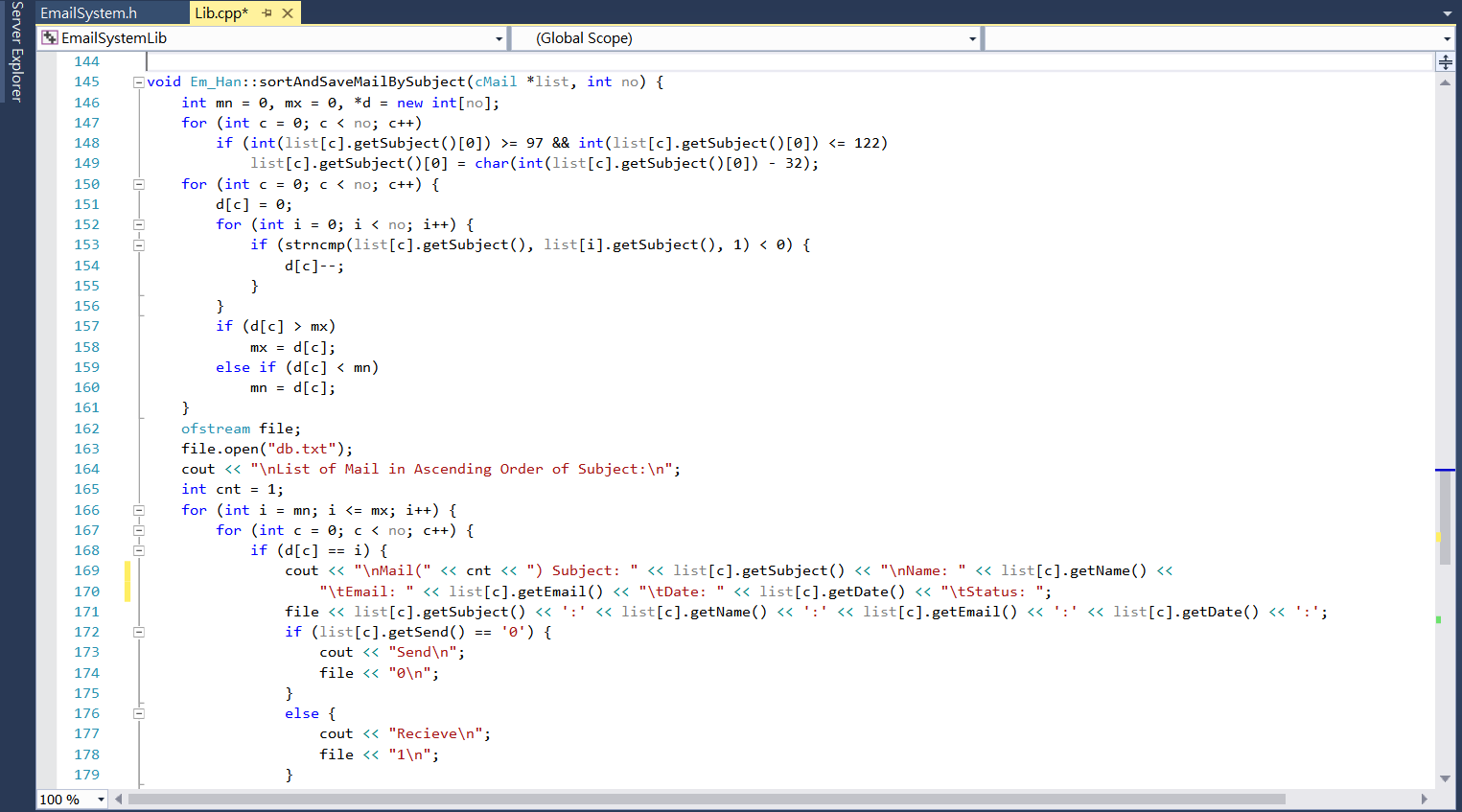
#Subject#:#Name#:#Email#:#Date#:#Send-Flag#

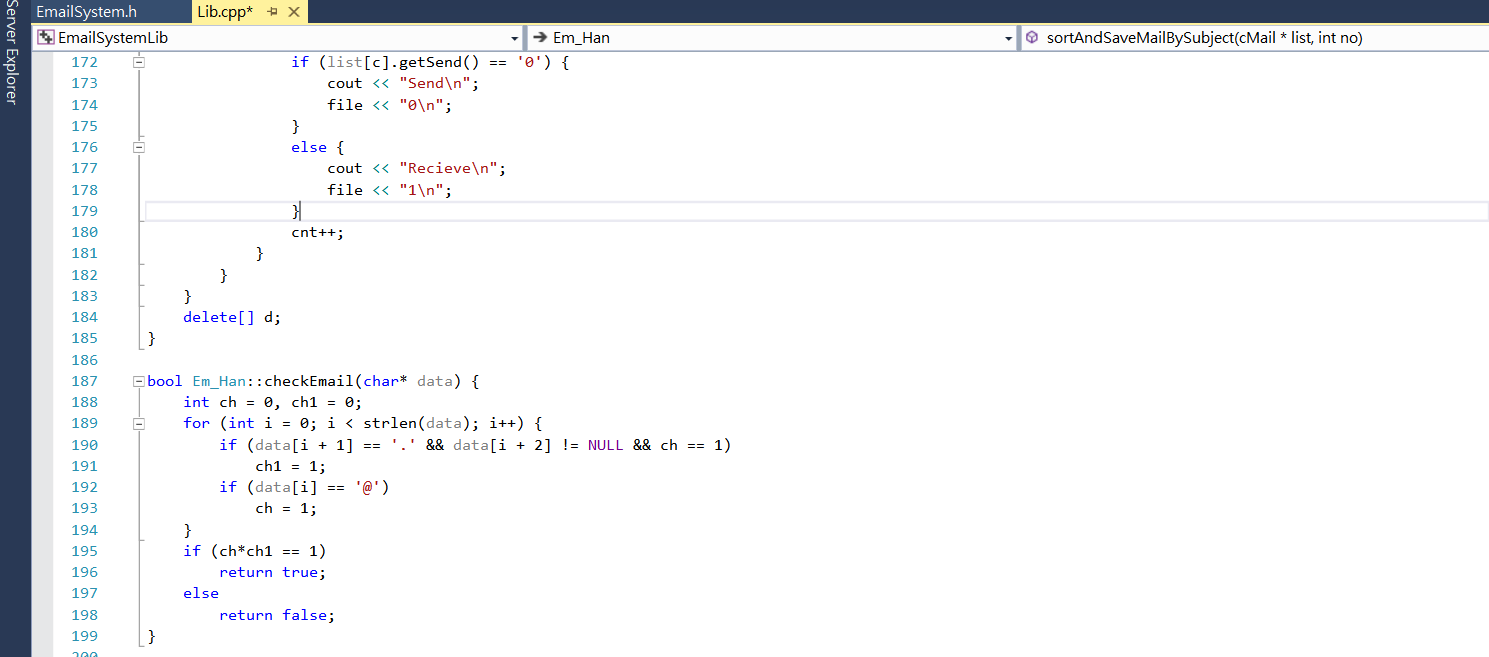
* Em\_Han::sortAndSaveMailBySubject()

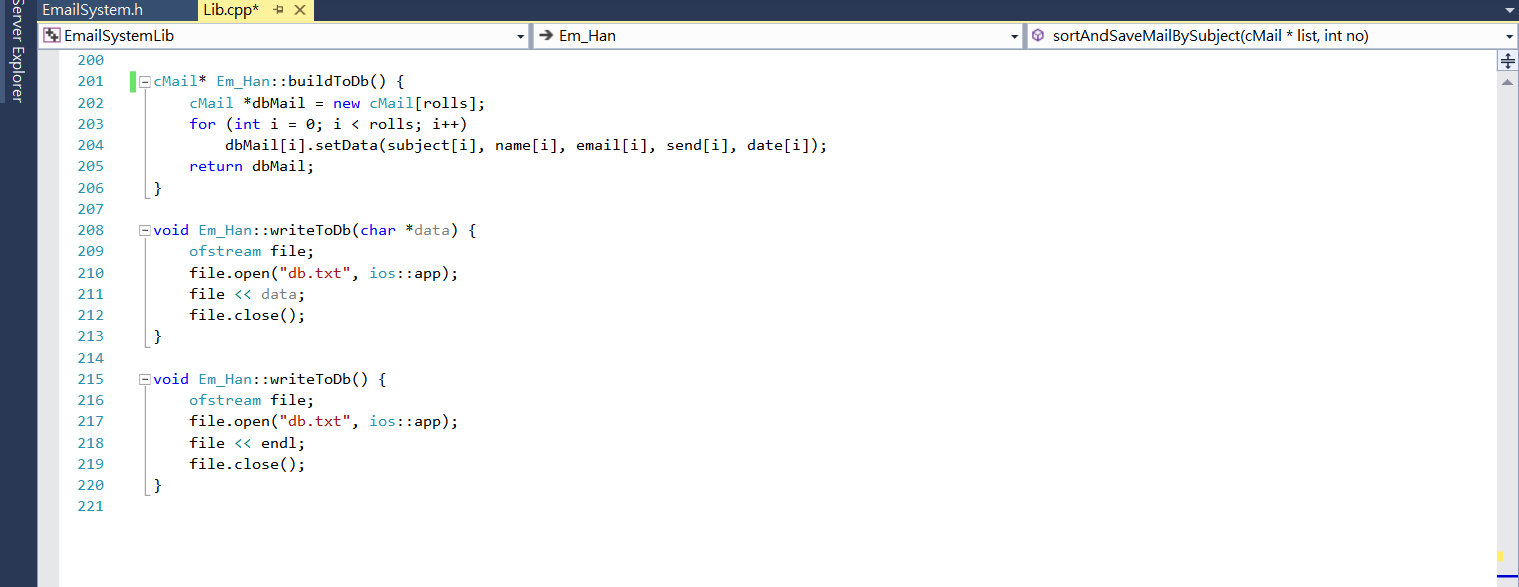
This function will take in a array of cMail object and sort them in ascending order of the Subject, in the process, it will convert the first character of the Subject into Upper Case Letter. After sorting the data, the function will display the sorted list and save the arranged datalist into the database, overwrites the previous stored data.



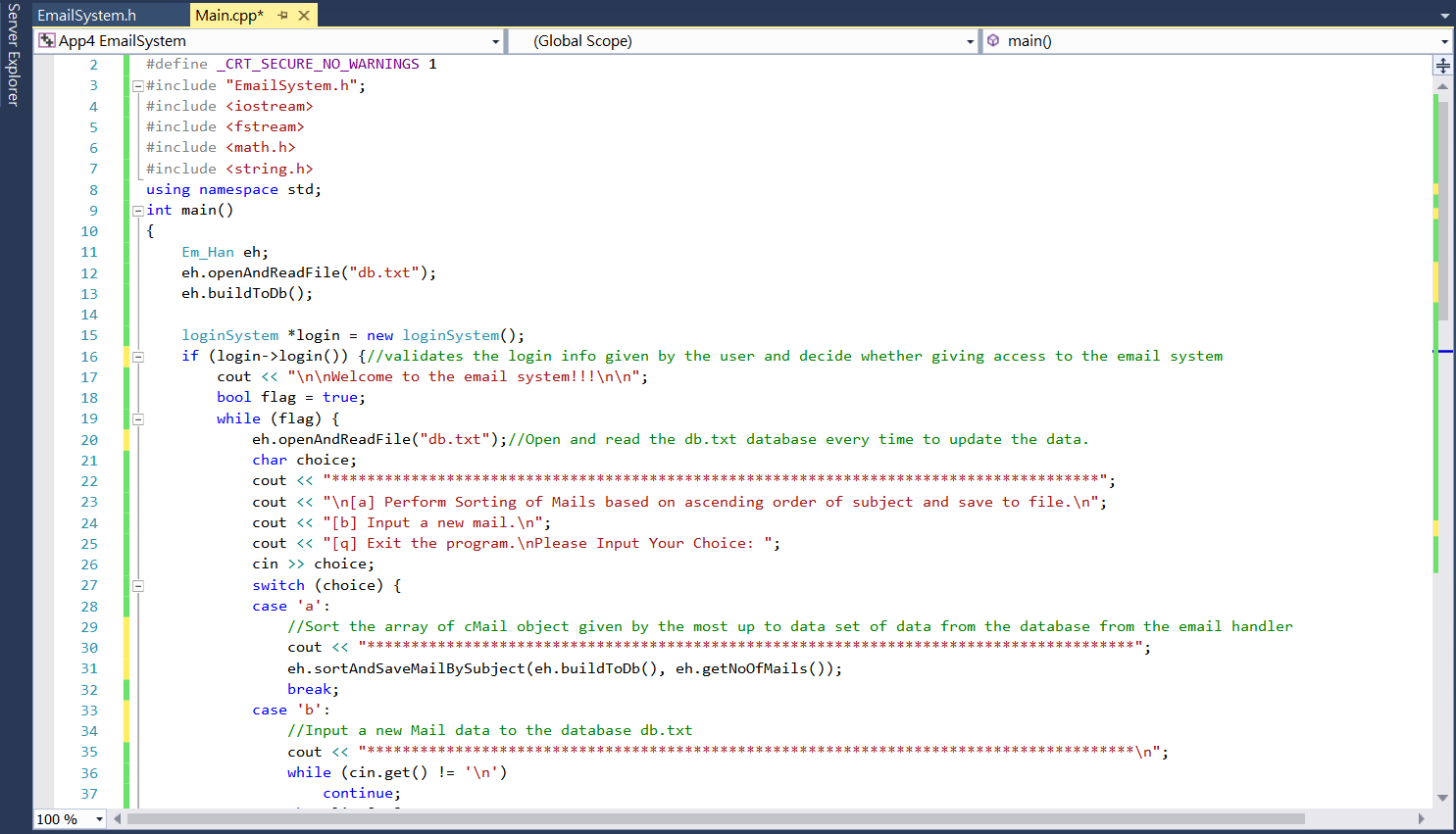


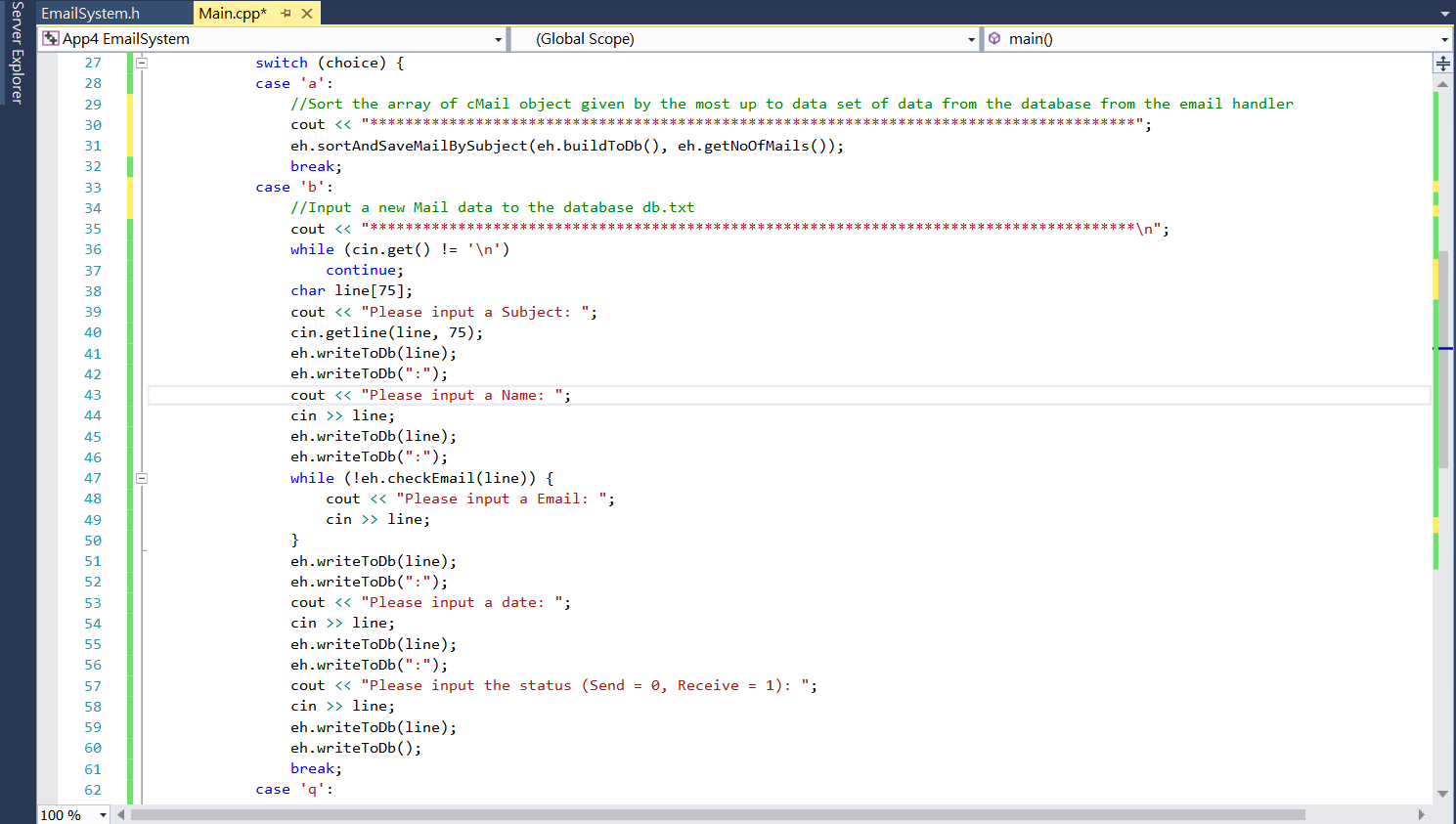


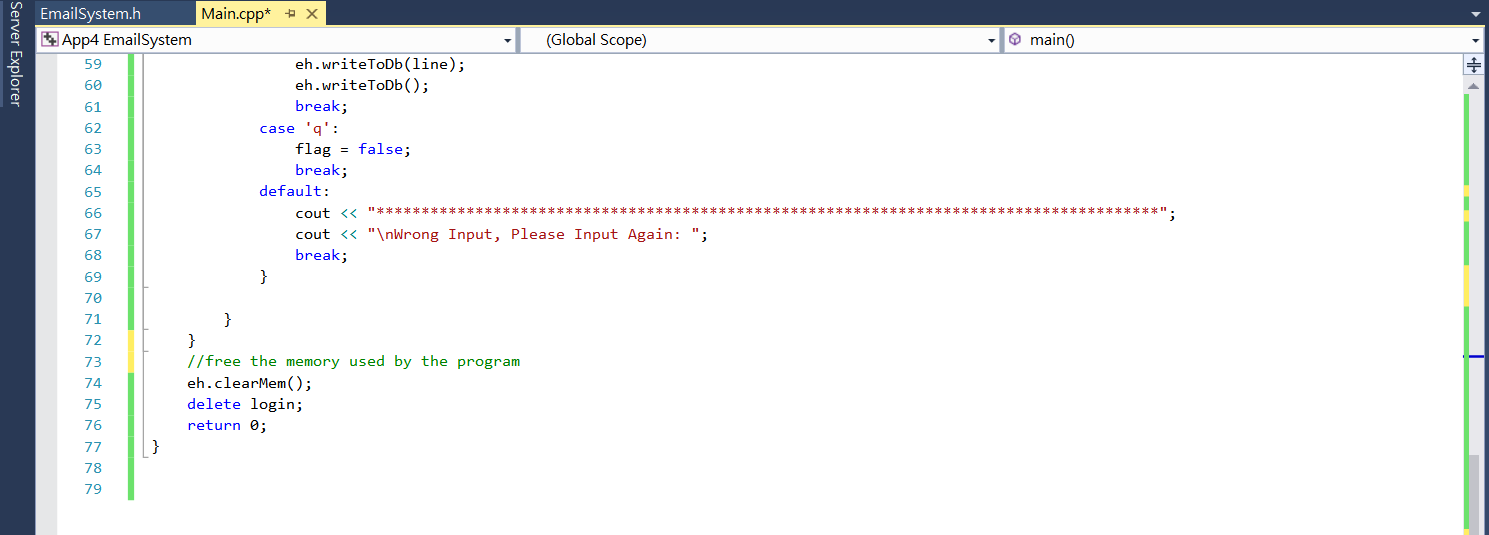




Main function👇







* + The main function represents the pipeline of the entire email system, which containing 3 options, the first options will sort the mail from the database and display a sorted list in the cmd, and overwrites the old database data.
  + The description for the pipeline A:

1. Email Handler updates the arrays containing the variables of all prototype cMail objects.
2. Email Handler calls function and retrieve an updated array of cMail from the database, then sort them by the sorting function, and updates the database.

* The description for the pipeline B:

1. Email Handler updates the arrays containing the variables of all prototype cMail objects.
2. The main function ask for inputs for a new mail object and create a new element in all array within the email handler, represents a new prototype cMail object.

**iv. Problem**

In this project,we have encounter three major difficulties

* First is the input file error
  + When the program read the character ‘0’ from a file
  + The program will return 0^2222
  + We fixed it by replacing ‘0’ to NULL character
* Second is the output file error
  + When we try to sort the subject after the user input a new subject the inputted data doesn’t save immediately
  + The error is caused by fail to close the file after reading.
* Third is the input data error
  + Once we input a wrong format data into the database
  + The program immediately saved the data into the file
  + The program input the wrong data to the database
  + Return a garbled message

To deal with the difficulties, we have two debugging methods

* One is using debug mode to check the error one by one
  + Using debug mode can reduce the probability that may break a the rogram
  + Search the Internet for solution by checking the error message
  + Rebuild the solution and use debug mode to repeat the steps
* Anotherone is using “cout<< *xxx*<<endl” to check the value of *xxx*
  + Sometimes error occur due to the wrong value of output
  + With the aids of debug, check the value of the wrong output value one by one
  + Re-modified the function with wrong output data and rebuild the solution

**v. Testing**

After building a solution, we still need to test the program to ensure the output is in expected.

The testing of our program include two main process

1. Input correct data

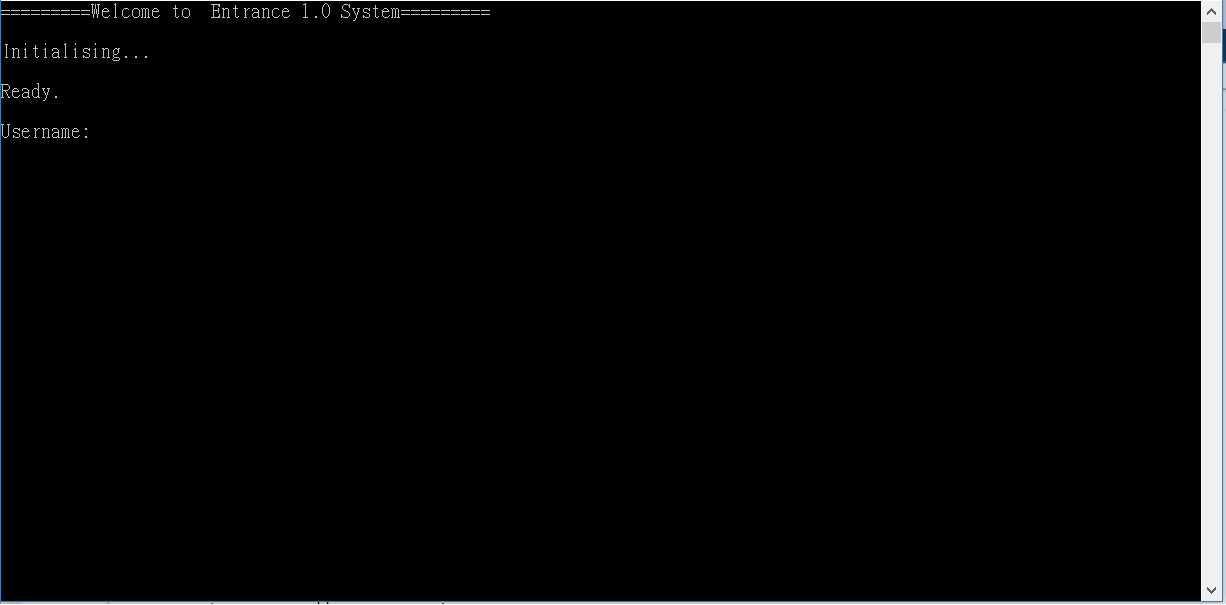
To check any run-time error. Sometimes the solution is successfully build, but when we input some data, the programming will give a run-time error. Hence, after successfully build the program, we will go throw all the flow of the program and check the file output to make sure that the program is well-functioned.

1. Input incorrect data

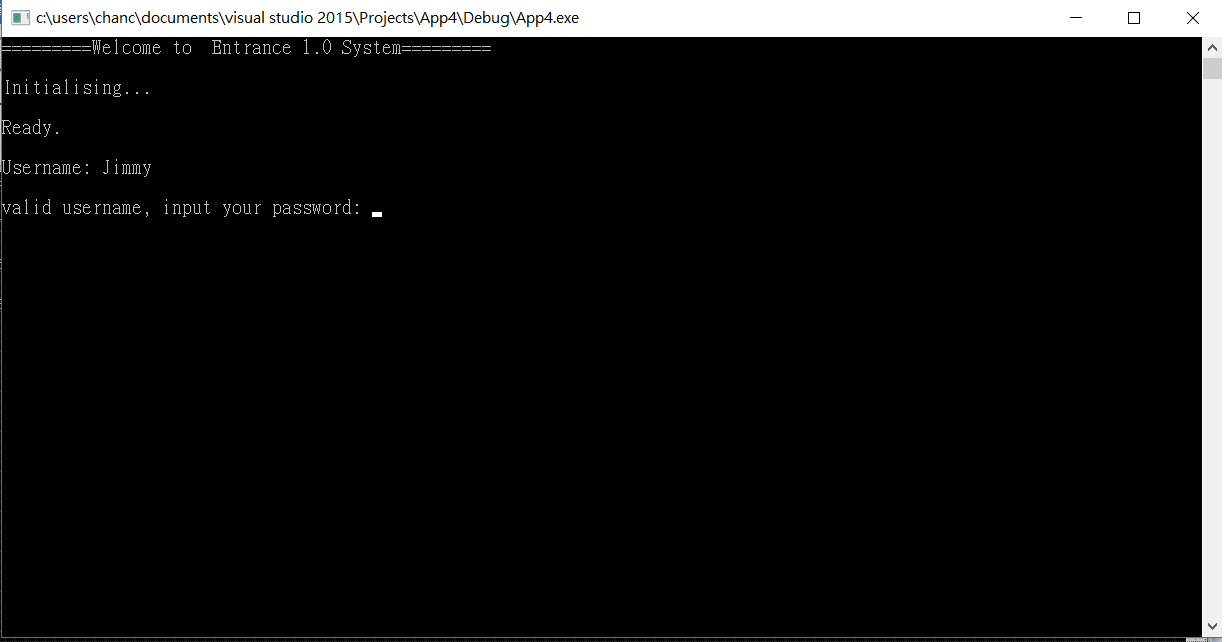
To check any garbled message. After building a well-functioned program, we need to consider that if the user input a unexpected data, what will the program display. To deal with the unexpected input data, we have set some restriction for the data input (such as checking the email “@”). We will input an incorrect data and see whether the program will ask us to re input again. This can make sure that the user input is inside our expectation, so as to prevent the program broken.

**3. Result**

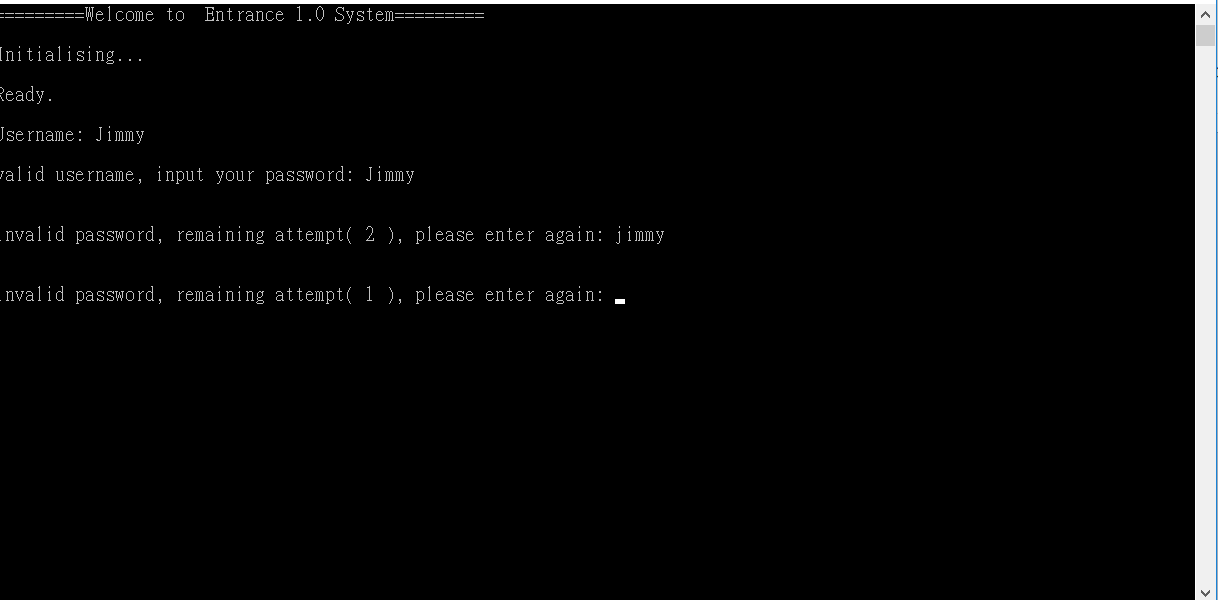
i. Login System👇



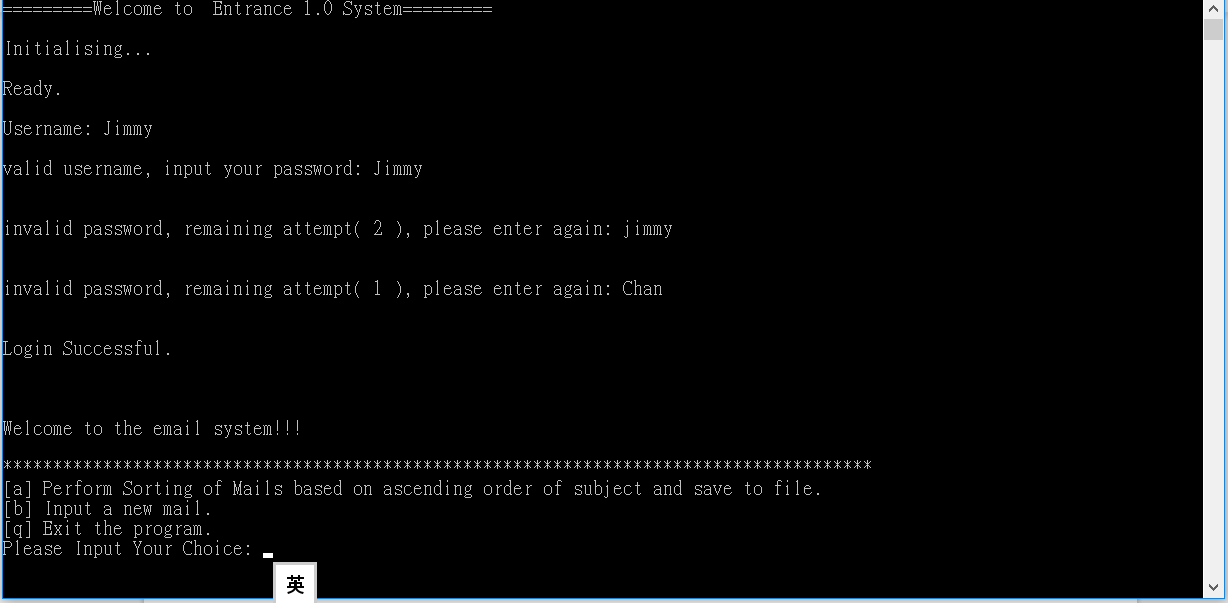
ii. Old Use need to enter a password👇



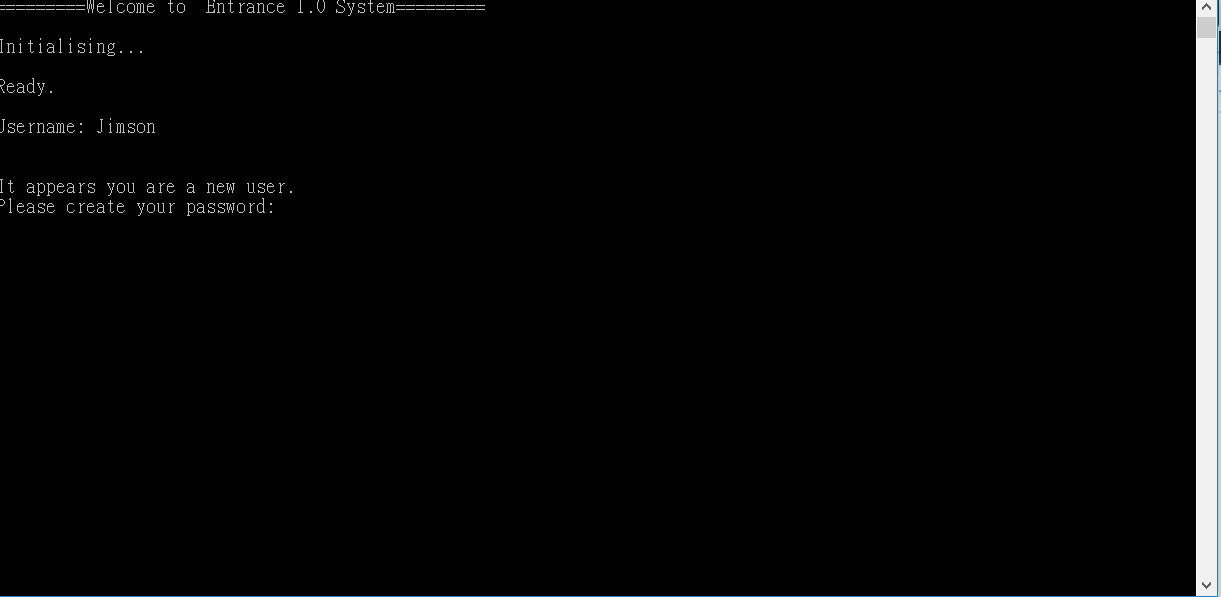
iii. Failure Login for entering a wrong password👇



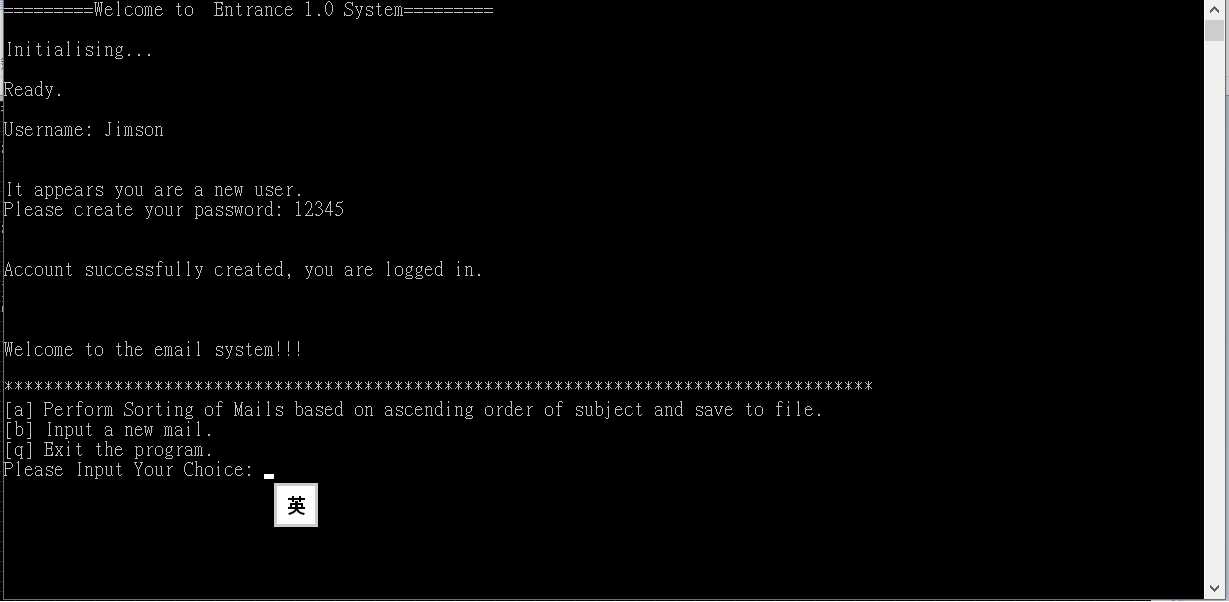
iv. Success login👇



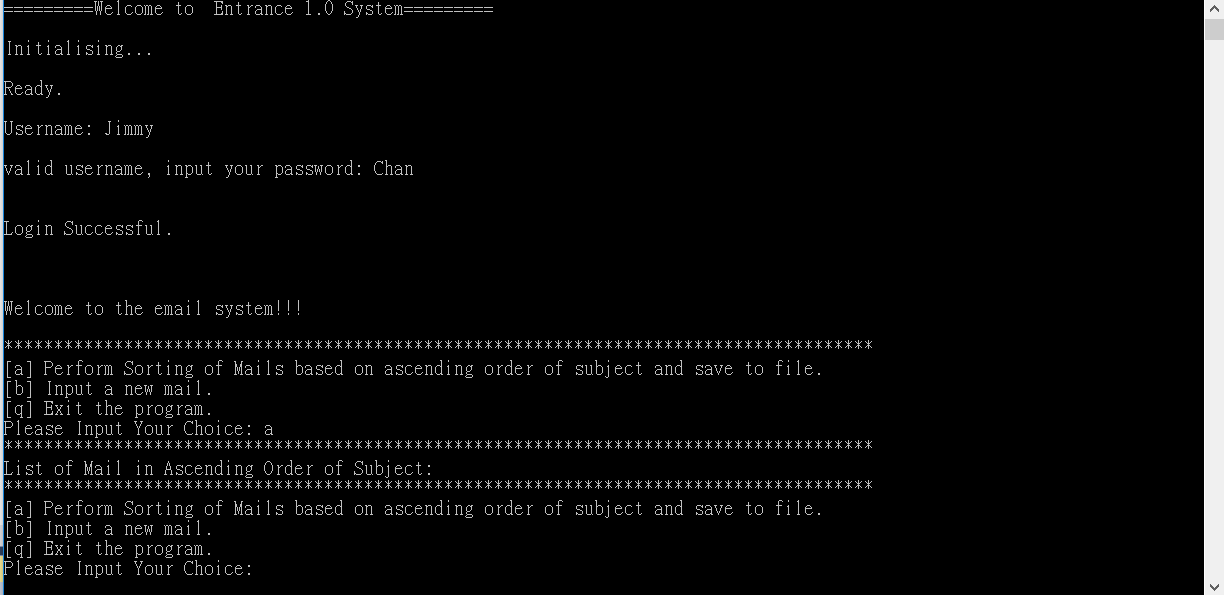
v. New user for setting password👇



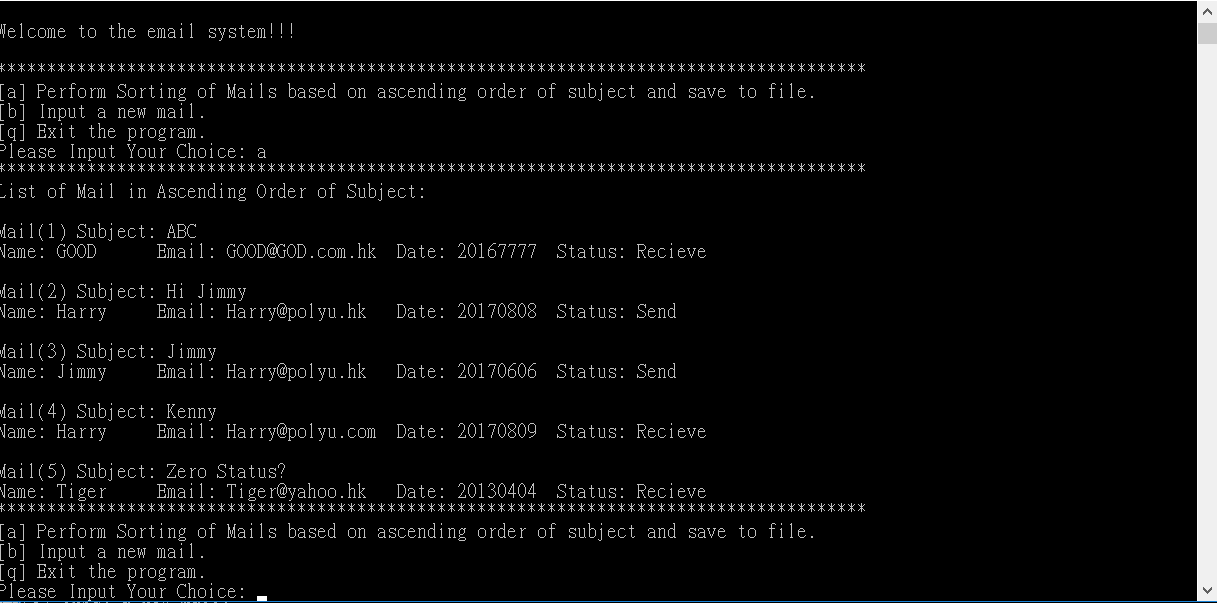
vi. Menu after successfully login👇



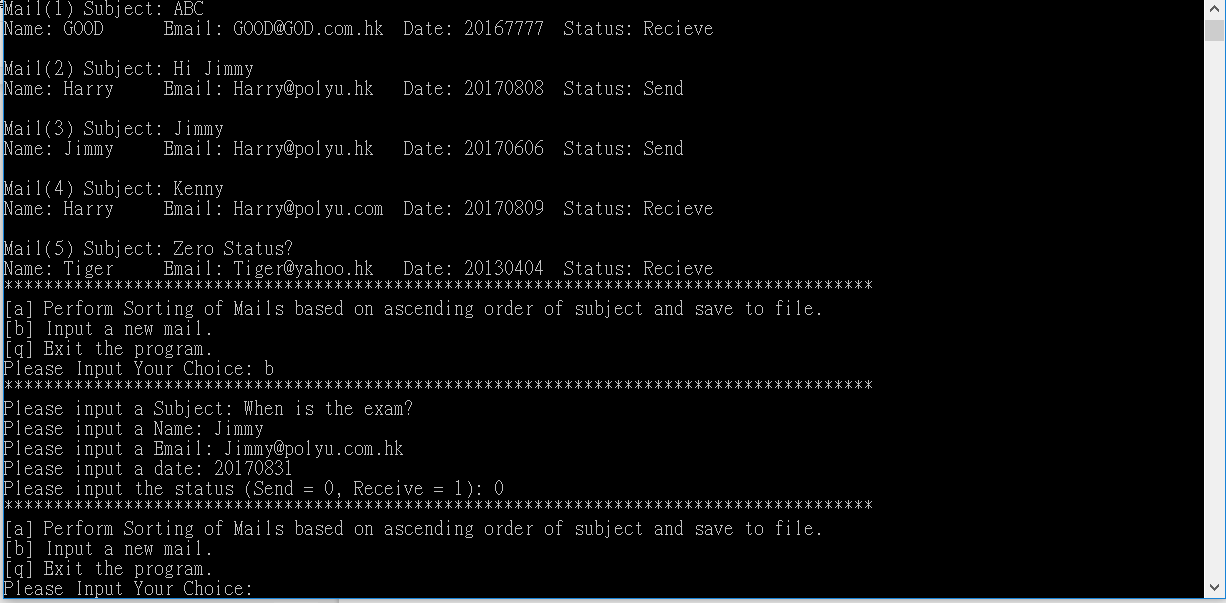
vii Sort a database from reading an empty file👇



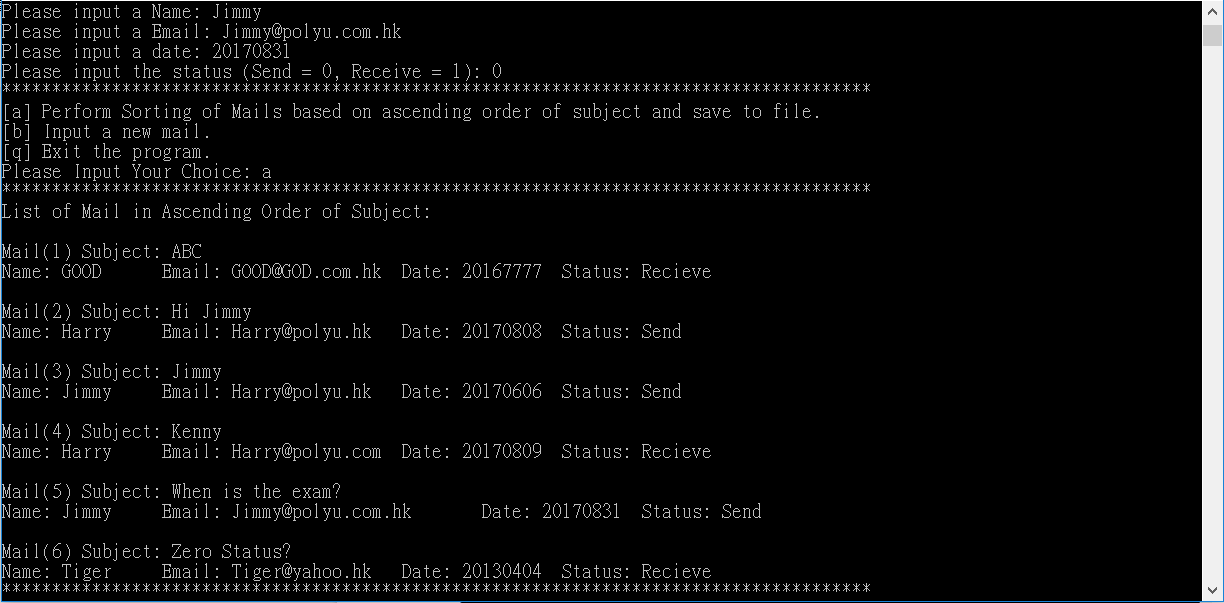
viii Sort a database from reading an edited file and save to files👇



ix. Input data into the database and save datas into an edited file👇



x. Sort after inputting the datas into database and save to file👇



**4. Conclusion**

**i. Feeling**

We have applied all the skills that taught in the lessons. We find that the application developement assignment is not difficult to handle, but it takes time to finish. At the beginning that we designed the flow of the program, to the middle that we argued for using what ways could be simpler and quicker, and to the final that we debug the program and tried to make the program perfect. We took nearly a week to finish this project.

Although we have learnt all of those syntax of the programming c++, but developing a program basing on the knowledge gained is not as easy as i have think. When I am developing the program, There is a lot of chance the program went into a error, which takes time to find and solve the problem. In another hand, developing the program is not only to make it work, we have to plan before we start on it.

After the assignment, i have learnt that developing a pipeline before working on the code is the most important thing to achieve a good program. Pipeline is important because it judge how the program runs, which also means the efficiency of the program. If we didn’t develope a proper pipeline, the program will become inefficient or made error happens. So to make a good program, I think developing a good pipeline before hand is the major thing to do.

**ii. Extension**

Although we think that our program has fulfilled the requirement stated, we still find that the program is not perfectly designed. We find that the restriction of the data input is not strictly restricted. For example, inputting the date of the program.

We design the date of the email as 8 digital number, since if we check the date year-by-year, month-by-month and day-by-day, the memory of the project will be a bit large. We have made a .cpp file which store the checking of the date of the program(checkDate.cpp). Yet, we did not add it into our program, as we tested that the function of checking date will lower the speed of processing the program.

In the future, we may design to build a own library which store the function of checking date, email, subject,etc. In this case, once we want to check those object, we can just implement the library. That will reduce a lot of time.