XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS102TC Programming with C++**
School Title	**School of Artificial Intelligence and Advanced Computing**
Assignment Title	**Coursework 2 (CW 2)**
Submission Deadline	**5 pm China time (UTC+8 Beijing) on Fri. 3rd. Nov. 2023**
Final Word Count	**NA**
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	**yes**

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	**2254891** **2254164** **2255970** **2251977** **2255470**

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved （F/P/M/D） (please modify as appropriate)**	**Final Score**

<table>
<tr><td></td><td></td><td>A</td><td>B</td><td>C</td><td></td></tr>
<tr><td>1st Marker – red pen</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">Moderation

– green pen</td><td>IM
Initials</td><td colspan="3">The original mark has been accepted by the moderator (please circle as appropriate):</td><td>Y / N</td></tr>
<tr><td></td><td colspan="3">Data entry and score calculation have been checked by another tutor (please circle):</td><td>Y</td></tr>
<tr><td>2nd Marker if needed – green pen</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">For Academic Office Use</td><td colspan="4">Possible Academic Infringement (please tick as appropriate)</td></tr>
<tr><td>Date Received</td><td>Days late</td><td>Late Penalty</td><td colspan="2">☐ Category A</td><td rowspan="5">Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____</td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td><td rowspan="4"></td><td colspan="2">☐ Category B</td></tr>
<tr><td colspan="2">☐ Category C</td></tr>
<tr><td colspan="2">☐ Category D</td></tr>
<tr><td colspan="2">☐ Category E</td></tr>
</table>

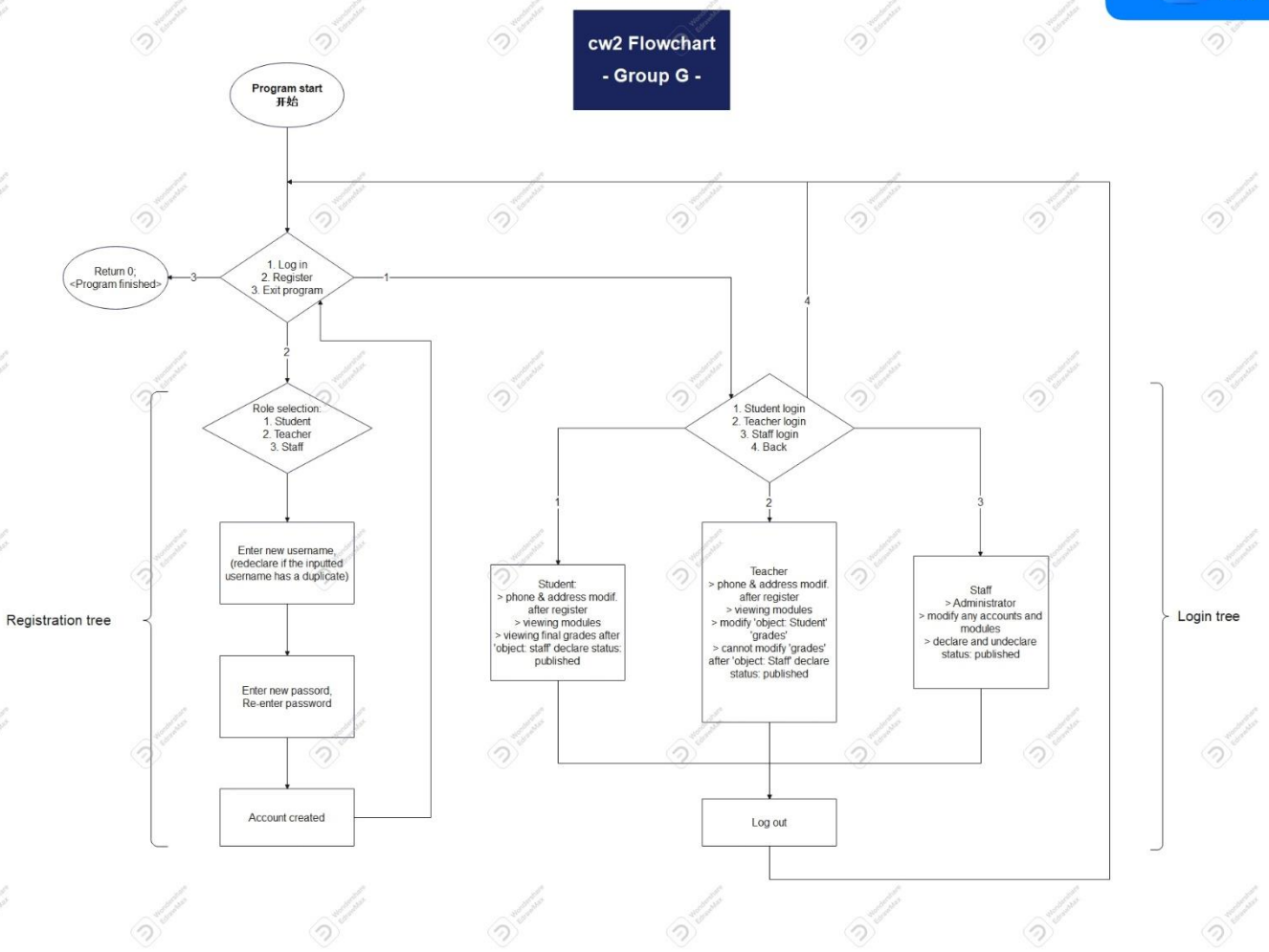
Students

(Please modify where necessary)

The assignment must be typed in an MS Word and converted to a PDF document. The document must be submitted via Learning Mall Online to the correct drop box. Only electronic submission is accepted and no hard copy submission.

All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.

1. Program Design



(Part a. and b. below will explain the flowchart in depth)

a) ***Flowchart for normal operation of all objects.***

Define data structures and classes:

- The ‘Student’ class is present with attributes such as account, password, phone number, address, enrolled module and score. This class has functions implemented for methods for login, modification of personal information, viewing elective module lists, and checking final grades.
- The ‘Teacher’ class is present with attributes such as account, password, phone number, address, and enrolled module. Have authorities for login, modification of personal information, viewing assigned module lists, and entering and modifying student grades while the status published is false.
- The ‘Admin’ (or staff) class is also present with account, password, phone number, and address attributes. Implemented functions for login; adding, modifying, and deleting every user's information; create modules and assigning users to a specified module; toggles the status published.

Implement the login interface:

Create a login interface that allows students, teachers, and staff to enter their account credentials and authenticate themselves.

Student functionalities

After successful login, display the student's homepage:

- On the homepage, allow students to modify their phone numbers and addresses.
- Students should be able to view a list of available modules and select one to check the final grade (if staff successfully assign any).

Teacher functionalities

After successful login, display the teacher's homepage:

- On the homepage, allow teachers to modify their phone numbers and addresses.
- Teachers should be able to view a list of assigned modules and select one to enter and modify student grades.
- Once the grades are published, teachers should not be allowed to modify them.

Staff functionalities

After successful login, display the staff's homepage:

- On the homepage, allow staff to modify their phone numbers and addresses.
- Staff should be able to add, modify and delete user information.
- Staff should be able to add, modify and delete module information.
- Staff should be able to post or retrieve module grades on the module interface.

b) ***Process design of abnormal operation of all objects.***

Disregarding the registration process that creates the user.txt and module.txt files would eventually cause the system and all objects to run abnormally.

2. Program Implementation

Complete the program implementation using C++ language based on the program design. This sub-task covers:

a) code implementation of login functions for different objects.

account.cpp:

```
#include"account.h"

Account::Account(int m_account, string m_password, int m_id)
{
	account = m_account;
	password = m_password;
	id = m_id;
}
void Account::display()
{
	if (id == 1)
	{
		cout << "\tStudent number：" << account << '\n';
		cout << "\tpassword：" << password << '\n';
	}
	else
		if (id == 2)
		{
			cout << "\tTeacher account：" << account << '\n';
			cout << "\tpassword：" << password << '\n';
		}
		else
			if (id == 3)
			{
				cout << "\tAdmin account：" << account << '\n';
				cout << "\tpassword：" << password << '\n';
			}
}
```

people.cpp (Starts from line 839)

```
	}
	else
	{
		char  c;
		number--;
```

```
                cout << "That's three times. There's still time." << number << "second";
                cout << "Wrong password, do you want to re-enter it?. . .(y/n) " << endl;
                cin >> c;
                if (number == 0) {
                cout << "Wrong password for locking" << '\n';
                cout << "Unable to re-enter password" << '\n';
                _getch();
                system("cls");
                break;
                }
                //cout << "That's three times. There's still time." << number << "second";
                system("pause");
                if (c == 'n')
                {
                system("cls");
                return;
                }
                system("cls");

        }
}
}
//register function
void Register::reg_user()//"register user" obtain the inputted username, will pass on to
'reg_pass' after completion
{
    system("cls");
    cout << "\nPlease enter your new username: ";
    cin >> user_name;
    //line: 994 - 1005 will filter if the inputted username is identical any other usernames
in the database
    ifstream username_check;
    username_temp = user_name + ".txt";
    cout << username_temp << '\n';
    username_check.open(username_temp.c_str());
    getline(username_check, username);
    getline(username_check, username);
    cout << username << '\n';
    if (user_name == username)
    {
        cout << "Username: " << user_name << " has been taken. Please enter a different
username." << '\n';
```

```
            system("pause");
            reg_user();
        }
        username = user_name;
        cout << "\nUsername - \"" << username << "\"\nUsername cannot be rename after
confirmation. Confirm? \n\n[1] Yes\n[2] No" << '\n';
        cin >> confirm;
        switch (confirm)
        {
        case '1'://continue for password making after user gives consent to the inputted username
            reg_pass();
            break;
        case '2'://in case user disagree with their name selection
            cin.clear();
            cin.ignore(10000, '\n');
            reg_user();
            break;
        default:
            cout << "Invalid input. Please try again" << '\n';
            cin.clear();
            cin.ignore(10000, '\n');
            reg_user();
            break;
        }
}

void Register::reg_pass()//"register password" obtain the inputted password, will pass on to
'reg_WriteAcc' after completion
{
    People_Manager censor;
    cout << "Please enter your new password:" << '\n';
    censor.pass_au(password);
    cout << "Please re-enter your new password:" << '\n';
    censor.pass_au(re_password);
    if (password == re_password)
    {
        cin.clear();
        cin.ignore(10000, '\n');
        reg_WriteAcc();
    }
    else if (password != re_password)// if the new password and confirmation password do not
match
```

```
    {
        cout << "Entered password is not the same!" << '\n';
        reg_pass();
    }
}

void Register::reg_role()//"register role" a role decider based on user input
{
    std::cout << "-------------------------" << std::endl;
std::cout << "     Role selection      " << std::endl;
std::cout << "-------------------------" << std::endl;
std::cout << "                         " << std::endl;
std::cout << "   [1] Student           " << std::endl;
std::cout << "                         " << std::endl;
std::cout << "   [2] Teacher           " << std::endl;
std::cout << "                         " << std::endl;
std::cout << "   [3] Staff             " << std::endl;
std::cout << "                         " << std::endl;
std::cout << "-------------------------" << std::endl;
std::cout << "Please enter your role: ";
std::cin >> confirm;  switch (confirm)
    {
    case '1':
        role = "student";//assign the student role to the newly created account
        reg_user();
        break;
    case '2':
        role = "teacher";//assign the teacher role
        reg_user();
        break;
    case '3':
        role = "admin";//assign the staff role
        reg_user();
        break;
    default:
        cout << "Invalid input. Please try again" << '\n';
        cin.clear();
        cin.ignore(10000, '\n');
        system("pause");
        system("cls");
        reg_role();
        break;
```

```
    }
1.      }
```

ii) code implementation of personal information operation functions for different objects. (5 marks)

student:

```
void People_Manager::studentOperation()//operation
{
    int inputAC;
    string inputPD;
    while (true)
    {
        cout << "Enter your ID:" << '\n';
        cin >> account;
        cout << "Enter your password:" << '\n';
        pass_au(inputPD);
        int flag = 0;
        int index;
        for (int i = 0; i < Acc_Num; i++)
        {
            if (M_Account[i]->AC == inputAC && M_Account[i]->PD == inputPD && M_Account[i]->id
== 1)
            {
                flag = 1;
                index = i;
                cout << "Login Successful" << '\n';
                system("pause");
                system("cls");
                break;
            }
        }
        if (flag)
        {
            int choice;
                 while (true)
         {
    cout << "-----------------------------" << endl;
    cout << "|1. View Personal Information              |" <<endl;
    cout << "-----------------------------" << endl;
    cout << "|2. change your password                  |" << endl;
    cout << "-----------------------------" << endl;
```

```
    cout << "|0. return                        |" << endl;
    cout << "-------------------------------" << endl;
    cin >> c;
switch (c)
{
case 1:
    for (int k = 0; k < Stu_Num; k++)
    {
        if (M_Student[k]->num == account)
        {
            M_Student[k]->display();
            system("pause");
            system("cls");
            break;
        }
    }
    break;

case 2:
    cout << "Enter the new password:" << '\n';
    cin >> password;
    M_Account[j]->password = password;
    account_save();
    cout << "Modified successfully" << '\n';
    system("pause");
    system("cls");
    break;

case 0:
    system("cls");
    return;
    break;
}

if (!flag)
{
    cout << "incorrect password" << '\n';
    cout << "Whether to re-enter....(y/n)" << '\n';
    cin >> b;
    if (b == 'n')
    {
        system("cls");
```

```
        return;
    }
    system("cls");
}
```

Teacher:

```
                case 3:
    cout << "Enter the new password:" << '\n';
    cin >> password;
    M_Account[j]->password = password;
    account_save();
    break;

case 0:
    system("cls");
    return;
}

system("cls");

if (!flag)
{
    char b;
    cout << "incorrect password" << '\n';
    cout << "Whether to re-enter....(y/n)" << '\n';
    cin >> b;
    if (b == 'n')
    {
        system("cls");
        return;
    }
    system("cls");
}
```

Staff:

```
                        case 4:
                            for (int ii = 0; ii < Acc_Num; ii++)
                            {
                                if (M_Account[ii]->account == ACCOUNT && M_Account[ii]->id
== 3)
                                {
```

```
                cout << "Enter the new password:" << '\n';
                cin >> password;
                M_Account[ii]->password = password;
                break;
            }
        }
        account_save();
        cout << "Successful change..." << '\n';
        system("pause");
        system("cls");
        break;
```

iii) code implementation staff's operation function on user information table. (5 marks)

```
void People_Manager::administratorOperation()
{
    string adminAccount;
    string adminPassword;

    cout << "Please enter an account number:" << '\n';
    cin >> adminAccount;
    cout << "Please enter your password:" << '\n';
    pass_au(adminPassword);

    int adminFlag = 0;
    for (int i = 0; i < Acc_Num; i++)
    {
        if (M_Account[i]->account == adminAccount && M_Account[i]->id == 3 &&
M_Account[i]->password == adminPassword)
        {
            adminFlag = 1;
            break;
        }
    }

    if (adminFlag)
    {
        cout << "Login Successful...." << '\n';
        system("pause");
        system("cls");
```

```
       int mainChoice = -1;
while (mainChoice != 0)
{
    cout << "--------------------" << '\n';
    cout << "|1. Student Information Operations     |" << '\n';
    cout << "--------------------" << '\n';
    cout << "|2. Account Operations     |" << '\n';
    cout << "--------------------" << '\n';
    cout << "|0. Log out     |" << '\n';
    cout << "--------------------" << '\n';
    cout << "Enter your choice:" << '\n';
    cin >> mainChoice;
    system("cls");

    if (mainChoice == 1)
    {
        int studentChoice = -1;
        while (studentChoice != 0)
        {
          std::cout << "------------------" << '\n';
std::cout << "|1. Add Student   |" << endl;
std::cout << "------------------" << '\n';
std::cout << "|2. Delete Student   |" << '\n';
std::cout << "------------------" << endl;
std::cout << "|3. Modify Student Information|" << '\n';
std::cout << "------------------" << '\n';
std::cout << "|4. Find Student  |" << endl;
std::cout << "------------------" << '\n';
std::cout << "|5. Display Student Information|" << '\n';
std::cout << "------------------" <<endl;
std::cout << "|6. Sort Students    |" << '\n';
std::cout << "------------------" << endl;
std::cout << "|0. Return      |" << '\n';
std::cout << "------------------" << '\n';
std::cout << "Enter your choice:" << endl;
std::cin >> studentChoice;            if (studentChoice == 1)
            {
                addStudent();
            }
            else if (studentChoice == 2)
            {
                deleteStudent();
```

```
            }
            else if (studentChoice == 3)
            {
                modifyStudent();
            }
            else if (studentChoice == 4)
            {
                findStudent();
            }
            else if (studentChoice == 5)
            {
                displayStudent();
            }
            else if (studentChoice == 6)
            {
                sortStudents();
            }
            else if (studentChoice == 0)
            {
                system("cls");
            }
        }
    }
}
    case 2:
        cout << "Account Operation" <<  '\n';

        break;

    case 0:
        cout << "Logging out..." <<  '\n';
        return;
    }
}
```

iv) code implementation staff's operation function on module information table. (5 marks)

```
    while (true)
{
    Account** newAccounts = new Account * [Acc_Num + 1];
std::cout << "-----------------" << std::endl;
std::cout << "|1. Add teacher account|" << std::endl;
std::cout << "-----------------" << std::endl;
std::cout << "|2. Delete teacher accounts|" << std::endl;
```

```
std::cout << "-----------------" << std::endl;
std::cout << "|3. View all accounts|" << std::endl;
std::cout << "-----------------" << std::endl;
std::cout << "|4. Change password    |" << std::endl;
std::cout << "-----------------" << std::endl;
std::cout << "|0. Logout          |" << std::endl;
std::cout << "-----------------" << std::endl;
std::cout << "Enter your choice:" << std::endl;
    cin >> choice;
    switch (choice)
    {
    case 1:
        while (true)
        {
            found = 0;

            cout << "Enter the employee number of the teacher:" << '\n';
            cin >> empNumber;
            for (int i = 0; i < Acc_Num; i++)
            {
                if (M_Account[i]->account == empNumber && M_Account[i]->id == 2)
                {
                    cout << "This teacher already exists, please re-enter..." << '\n';
                    found = 1;
                    break;
                }
            }
            if (!found)
            {
                break;
            }
        }
        for (int i = 0; i < Acc_Num; i++)
        {
            newAccounts[i] = M_Account[i];
        }
        Account* newAccount = new Account(empNumber, "111111", 2);
        newAccounts[Acc_Num] = newAccount;
        delete[] M_Account;
        M_Account = newAccounts;
        Acc_Num++;
        saveAccounts();
```

```
        cout << "Account added successfully" << '\n';
        system("pause");
        system("cls");
        break;
             case 2:
    cout << "Enter the employee number of the teacher:" << '\n';
    cin >> empNumber;
    for (int i = 0; i < Acc_Num; i++)
    {
        if (M_Account[i]->account == empNumber && M_Account[i]->id == 2)
        {
            for (int j = i; j < Acc_Num; j++)
            {
                M_Account[j] = M_Account[j + 1];
            }
            break;
        }
    }
    Acc_Num--;
    saveAccounts();
    cout << "Deletion successful..." << '\n';
    system("pause");
    system("cls");
    break;

case 3:
    for (int i = 0; i < Acc_Num; i++)
    {
        M_Account[i]->display();
    }
    system("pause");
    system("cls");
    break;

case 4:
    for (int i = 0; i < Acc_Num; i++)
    {
        if (M_Account[i]->account == ACCOUNT && M_Account[i]->id == 3)
        {
            cout << "Enter the new password:" << '\n';
            cin >> newPassword;
            M_Account[i]->password = newPassword;
```

```
            break;
        }
    }
    saveAccounts();
    cout << "Modification successful..." << '\n';
    system("pause");
    system("cls");
    break;

case 0:
    flag = 1;
    break;
}

if (flag)
{
    system("cls");
    break;
}
```

v) code implementation of the teacher's operation on the module information table. (5 marks)

```
if (flag)
{
    int choice;

   while (true)
{
    const char* menu = "╔═══════════════════════════════════════════╗\n"
        "║  1. View all student info  ║\n"
        "╠═══════════════════════════════════════════╣\n"
        "║  2. View individual info   ║\n"
        "╠═══════════════════════════════════════════╣\n"
        "║  3. Change password        ║\n"
        "╠═══════════════════════════════════════════╣\n"
        "║  0. Return                 ║\n"
        "╚═══════════════════════════════════════════╝\n";

    printf("%s", menu);

    cin >> choice;
}       int flag2 = 0;
```

```
        switch (choice)
{
case 1:
    for (int i = 0; i < Stu_Num; i++)
    {
        M_Student[i]->display();
    }
    system("pause");
    system("cls");
    break;

case 2:
    cout << "Enter the student's student number:" << '\n';
    cin >> studentNum;
    for (int i = 0; i < Stu_Num; i++)
    {
        if (M_Student[i]->num == studentNum)
        {
            found = true;
            M_Student[i]->display();
            system("pause");
            system("cls");
            break;
        }
    }
    if (!found)
    {
        cout << "Cannot find this person." << '\n';
    }
    break;

case 3:
    cout << "Enter the new password:" << '\n';
    cin >> newPassword;
    M_Account[j]->password = newPassword;
    saveAccounts();
    break;

case 0:
    system("cls");
    return;
    break;
```

```
}

system("cls");
```

vi) code implementation for students to view module information tables. (5 marks)

```
if (flag)
{
    int c;
    while (true)
    {
        const char* menu = "-----------------------------------\n"
            "|1. View your own information      |\n"
            "-----------------------------------\n"
            "|2. Change password                |\n"
            "-----------------------------------\n"
            "|0. Return                         |\n"
            "-----------------------------------\n";
        printf("%s", menu);
        if (c == 1)
{
    bool found = false;
    for (int k = 0; k < Stu_Num; k++)
    {
        if (M_Student[k]->num == account)
        {
            M_Student[k]->display();
            found = true;
            break;
        }
    }
```

vii) code quality covering naming rules of variables and comments of functions. (5 marks)

viii) object-oriented program development covering object definition and object encapsulation (5 marks)

main.cpp:

```
#include"global.h"
#include"people.h"

//sets of objects required for the system to work properly
People_Manager peo;
Register reg;
char ch;
bool quit1 = false, quit2 = false;

void regConfirm();//prototype function (the main 'reConfirm' will not work properly without this)

void menu_login()//login interface
{
    system("cls");
    cout << "-------------------------" << '\n';
    cout << "         Login           " << '\n';
    cout << "-------------------------" << '\n';
    cout << "                         " << '\n';
    cout << "   [1] Student Login     " << '\n';
    cout << "                         " << '\n';
    cout << "   [2] Teacher Login     " << '\n';
    cout << "                         " << '\n';
    cout << "   [3] Staff Login       " << '\n';
    cout << "                         " << '\n';
    cout << "   [4] Back              " << '\n';
    cout << "                         " << '\n';
    cout << "-------------------------" << '\n';
    cout << "Enter your choice: ";
    cin >> ch;
    switch (ch)
    {
    case '1':
        peo.stu_oper();
        break;
    case '2':
        peo.tea_oper();
        break;
    case '3':
        peo.man_oper();
```

```
            break;
        case '4':
            system("cls");
            quit2 = true;
            break;
        default:
            cout << "Invalid option entered! Please try again" << '\n';
            system("pause");
            system("cls");
            break;
        }
}

void menu_main()//main menu interface
{
    system("cls");
    cout << "---- LM_Retro v.0.5B ----" << '\n';
    cout << "  Copyright 2023 Team_G  " << '\n';
    cout << "-------------------------" << '\n';
    cout << "        Main Menu        " << '\n';
    cout << "-------------------------" << '\n';
    cout << "                         " << '\n';
    cout << "   [1] Login             " << '\n';
    cout << "                         " << '\n';
    cout << "   [2] Register          " << '\n';
    cout << "                         " << '\n';
    cout << "   [3] Exit Program      " << '\n';
    cout << "                         " << '\n';
    cout << "-------------------------" << '\n';
    cout << "Enter your choice: ";
    cin >> ch;
    switch (ch)
    {
    case '1':
        while (!quit2) {
            menu_login();
        }quit2 = false;
        break;
    case '2':
        regConfirm();
        break;
    case '3':
```

```
            system("cls");
            cout << "\nHave a great day!" << '\n';
            system("pause");
            quit1 = true;
            break;
        default:
            cout << "Invalid option entered! Please try again" << '\n';
            system("pause");
            break;
        }
}

void regConfirm()//confirmation if the user wants to register or not
{
    system("cls");
    cout << "This will open the Registration Menu. Continue?\n\n[1] Yes\n[2] No\n";
    cin >> ch;
    switch (ch)
    {
    case '1':
        system("cls");
        reg.reg_role();
        break;
    case '2':
        menu_main();
        break;
    default:
        cout << "Invalid option entered! Please try again" << '\n';
        system("pause");
        regConfirm();
        break;
    }
}

//executes the code
int main() {
    cout << '\n';
    while (!quit1) {
        menu_main();
    }
    return 0;
```

global.h:

```
#include<iostream>
#include<fstream>
#include<conio.h>
#include<string>
#define FILENAME1 "student.txt"
#define FILENAME2 "account.txt"
using namespace std;
```

people.h:

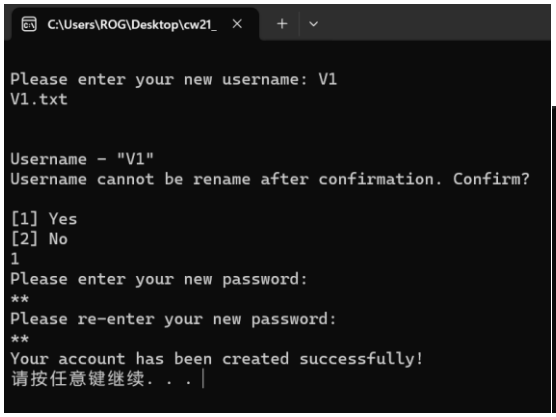
```
#include"student.h"
#include"account.h"

class People_Manager
{
public:

    int Stu_Num;//Number of students
    Student** M_Student;//Student array pointer

    void stu_init();//Initialize
    int stu_num();//Initial quantity

    int Acc_Num;//Account number
    Account** M_Account;//Account array pointer

    void acc_init();//Initialize
    int acc_num();//Initial quantity


    void account_save();//Account preservation
    void stu_add();//Add a student
    void stu_sort();//Sort student
    void stu_del();//Delete a student
    void stu_mod();//Modifying student
    void stu_find();//Find a student
    void stu_save();//Preserve students
```

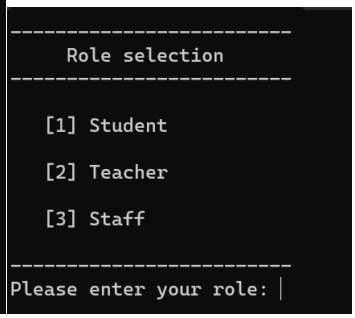
```
    void stu_show();//Display student
    void stu_oper();//Student operation
    void tea_oper();//Teacher operation
    void man_oper();//Admin operation
    int pass_au(string&);//Password censoring system
};

class Register
{
public:

    char confirm;
    char choice;
    string username, user_name, password, re_password, file_name, role, username_temp;

    void reg_WriteAcc();
    void reg_user();
    void reg_pass();
    void reg_role();
};
```

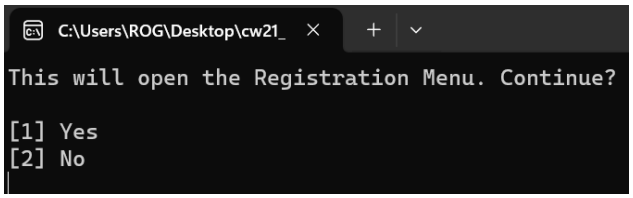
3. Program Testing (10 group marks)

Execute the developed system and complete the system function test.

i) normal operation testing of all objects. (8 marks)







ii) abnormal operation testing of all objects. (2 marks)