CS205 C/C++ Programming - Lab ASSIGNMENT 4

Name : 杨睦圳（Yang Muzhen）

SID： 11813010

Part 1 - Analysis

For q1, we need to generate the random data. We can define a function `random(int a, int b)` to get a random integer between a and b with the function `rand()`, which will provide a random integer. Everytime we generate a new id, check if it already exists and make sure we get all id distinct.

For q2, for every student we should check their all labs' scores and if there are more than two scores is zero, print its id out.

For q3, we can use `ofstream` to write the data into the .csv file.

For q4, read the data from the input file and when the data is illegal, give out the message and directly exit. Use a integer array to store the average score of each lab and get the total average score of them. Then find out the lab with average score less than the total average score.

For q5, define a function `trim()` to delete the whitespaces on both ends of the input. Use a string array to store all the commands and search for the corresponding index of the input command. If the input doesn't exist in the array, print out the message. The program will be case-sensitive.

Part 2 - Code

The program is worked on cygwin

Q1 Q2 Q3 is in one .cpp file

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <set>
using namespace std;

int random(int a, int b){
    int r = rand() % (b - a + 1);
    return a + r;
}


void showTheScores(int *ids, int** scores, int n){
    int i,j;
    for(i = 0; i < n; i++){
        cout << ids[i] << ": ";
        for(j = 0; j < 14; j++){
            cout << scores[j][i];
```

```
            if(j == 13)
                cout << endl;
            else
                cout << ", ";
        }
    }
    cout << endl;
}
void generateData(int *ids, int** scores, int n){
    int i,j;
    set<int> idset;
    for(i = 0; i < n; i++){
        ids[i] = random(2000,2020) * 10000 + random(0,9999);

        if(idset.count(ids[i])){
            //if the id already exists
            --i;
            continue;
        }
        idset.insert(ids[i]);

        for(j = 0; j < 14; j++)
            scores[j][i] = random(0,5);
    }

    showTheScores(ids,scores,n);
}
void printAbsenceTwice(int *ids, int** scores, int n){
    cout << "the SID of the students whose absent time equal or exceed 2: " <<
endl;
    int i,j,t;
    for(i = 0; i < n; i++){
        t = 0;
        for(j = 0; j < 14; j++){
            if(scores[j][i] == 0)
                t++;
        }
        if(t >= 2)
            cout << ids[i] << endl;
    }
    cout << endl;
}
void writeToCSV(string& file, int *ids, int** scores, int n){
    ofstream fout;
    fout.open(file);

    int i,j;
    for(i = 0; i < n; i++){
        fout << ids[i] << ",";
        for(j = 0; j < 14; j++){
            fout << scores[j][i];
            if(j == 13)
                fout << endl;
            else
                fout << ",";
        }
    }
```

```
    cout << "Successfully write to csv file!" << endl;
}
int main(){
    int n;
    srand((int)time(0));
    string fileName = "lab_records.csv";

    cout << "Please input the number of students: ";
    cin >> n;
    while(n <= 0 || n > 210000){
        cout << "The number should be a integer between 1 and 210000, please
input again: ";
        cin >> n;
    }

    int *scores[14];

    int *ids = new int[n];
    int i;
    for(i = 0; i < 14; i++)
        scores[i] = new int[n];

    //q1
    generateData(ids,scores,n);
    //q2
    printAbsenceTwice(ids,scores,n);
    //q3
    writeToCSV(fileName, ids,scores,n);

    for(i = 0; i < 14; i++)
        delete[] scores[i];
    delete[] ids;
}
```

Q4

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

void trim(string &s){
    s.erase(0,s.find_first_not_of(' '));
    s.erase(s.find_last_not_of(' ') + 1);
}

void printIllegal(string& in){
    cout << "There is a illegal input as below, program exits!" << endl;
    cout << in << endl;
}

bool change_info(string& a, vector<int>& ids, vector<int*>& scores){
    string in = a;
    trim(in);
    int* score = new int[14];

    //get the id in string
```

```
    int index = in.find(',');
    if(index == std::string::npos)
        return false;
    string temp = in.substr(0,index);
    in = in.substr(index + 1);
    trim(temp);
    if(temp.size() != 8)
        return false;

    //check id information
    int i, id = 0;
    for(i = 0; i < 8; i++){
        if(temp[i] > '9' | temp[i] < '0')
            return false;
        id = id * 10 + temp[i] - 48;
    }
    if(id / 10000 > 2020 | id / 10000 < 2000)
        return false;

    //check if the id already exist
    for(i = 0; i < ids.size(); i++){
        if(id == ids[i])
            return false;
    }
    ids.push_back(id);


    //get the 14 lab scores
    for(i = 0; i < 13; i++){
        index = in.find(',');
        if(index == std::string::npos)
            return false;
        temp = in.substr(0,index);
        trim(temp);
        in = in.substr(index + 1);
        if(temp.size() != 1 | temp[0] > '5' | temp[0] < '0')
            return false;
        score[i] = temp[0] - 48;
    }
    trim(in);
    if(in.size() != 1 | in[0] > '5' | in[0] < '0')
        return false;
    score[13] = in[0] - 48;

    scores.push_back(score);
    return true;
}

bool read_input(string& fileName, vector<int>& ids, vector<int*>& scores){
    ifstream fin(fileName);
    if(!fin.good()){
        cout << "File missing! Program exit!"<< endl;
        return false;
    }
    string in;
    while (getline(fin,in)){
        if(!change_info(in, ids, scores)){
            printIllegal(in);
```

```
            return false;
        }
    }
    return true;
}


int main(){
    vector<int> ids;
    vector<int*> scores;

    string fineName = "lab_records.csv";
    if(!read_input(fineName, ids, scores))
        return EXIT_SUCCESS;
    int n = ids.size();

    int i,j;
    auto *avg = new double[14];
    double total = 0;
    for(i = 0; i < 14; i++){
        avg[i] = 0;
        for(j = 0; j < n; j++)
            avg[i] += scores[j][i];
        avg[i] = avg[i] / n;
        total += avg[i];
    }
    total = total / 14;

    cout << "The id of the lab which have average score less than the total
average score is:" << endl;
    for(i = 0; i < 14; i++){
        if(avg[i] < total)
            cout << (i + 1) << endl;
    }

    delete[] avg;
    for(i = 0; i < n; i++)
        delete[] scores[i];
}
```

Q5

```
#include <iostream>
#include <algorithm>
#define START_CMD 0
#define STOP_CMD 1
#define RESTART_CMD 2
#define RELOAD_CMD 3
#define STATUS_CMD 4
#define EXIT_CMD 5
using namespace std;

void trim(string &s){
    s.erase(0,s.find_first_not_of(' '));
    s.erase(s.find_last_not_of(' ') + 1);
```

```
}

int main(){
    string in;

    string commands[] = {"start", "stop", "restart","reload","status",
"exit","invalid"};

    cout << "Program start!" << endl;

    while(getline(cin,in)){
        trim(in);
        auto address = find(commands + 0, commands + 6, in);
        int index = distance(commands + 0, address);

        switch(index){
            case START_CMD:
                cout << "command <start> recognized" << endl;
                break;
            case STOP_CMD:
                cout << "command <stop> recognized" << endl;
                break;
            case RESTART_CMD:
                cout << "command <restart> recognized" << endl;
                break;
            case RELOAD_CMD:
                cout << "command <reload> recognized" << endl;
                break;
            case STATUS_CMD:
                cout << "command <status> recognized" << endl;
                break;
            case EXIT_CMD:
                cout << "Programme successfully exit!" << endl;
                return 0;
            default:
                cout << "Invalid command" << endl;
                break;
        }

    }
}
```

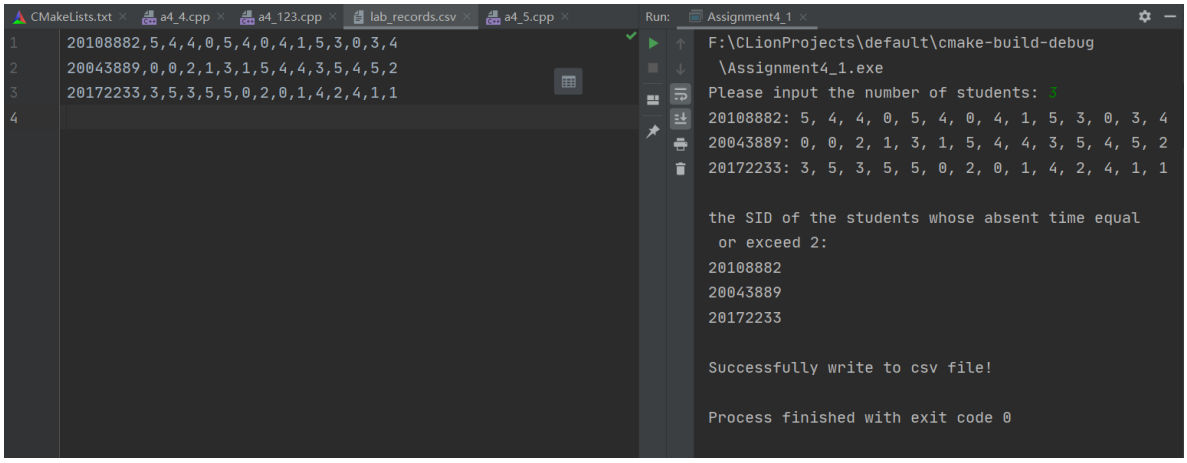
Part 3 - Result & Verification

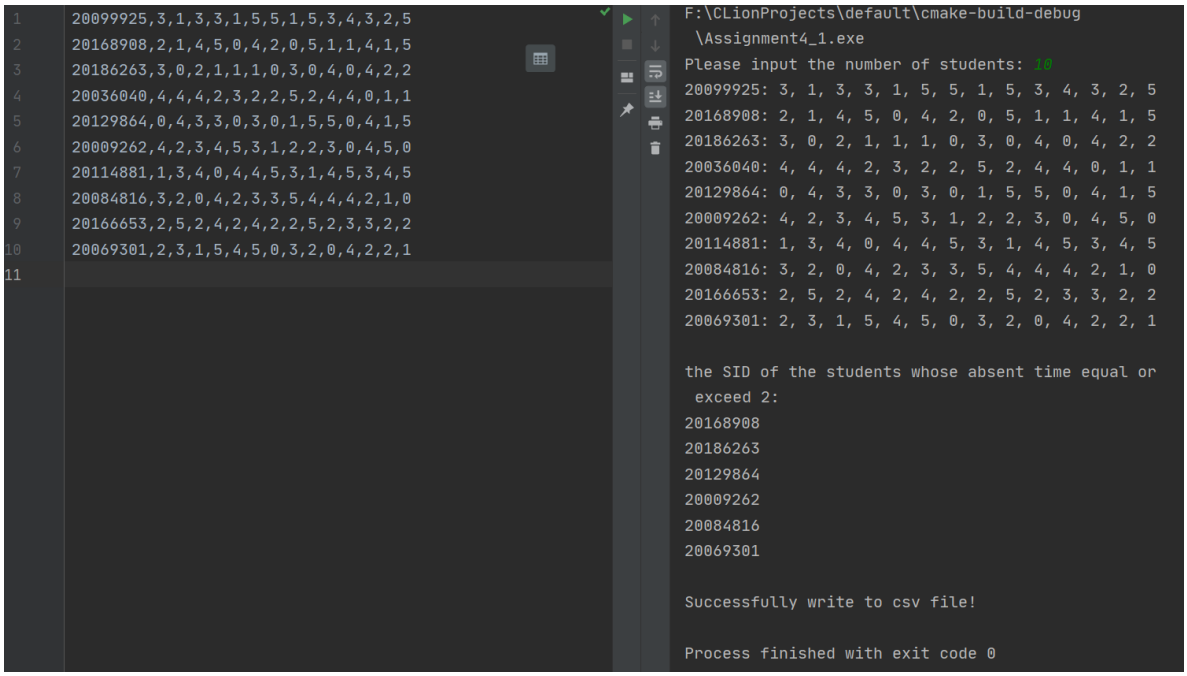
Compile

```
sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -c a4_123.cpp

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -o a4_123 a4_123.o

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -c a4_4.cpp

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -o a4_4 a4_4.o

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -c a4_5.cpp

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
$ g++ -o a4_5 a4_5.o

sdmms@DESKTOP-09CJ9E6 /cygdrive/c/users/sdmms/Desktop
```

Q1, Q2, Q3

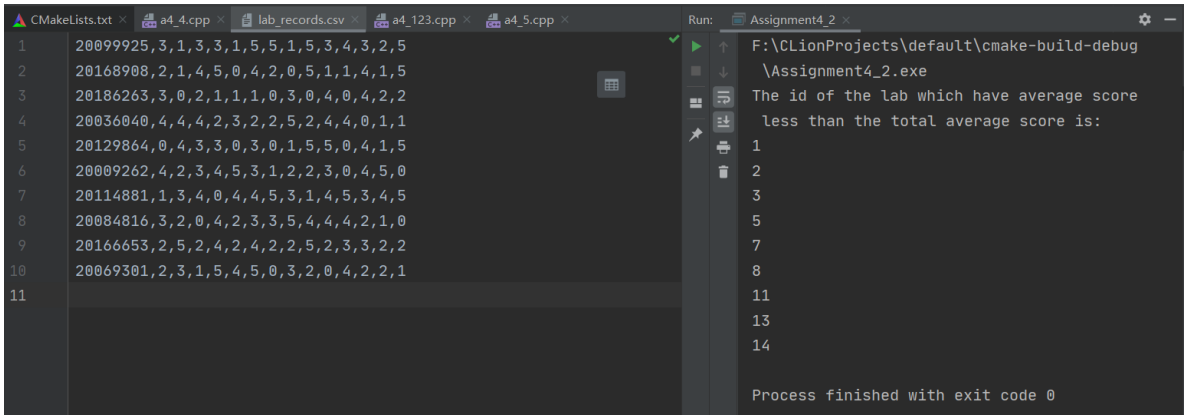
The data in the file after the program execute is shown at the left.

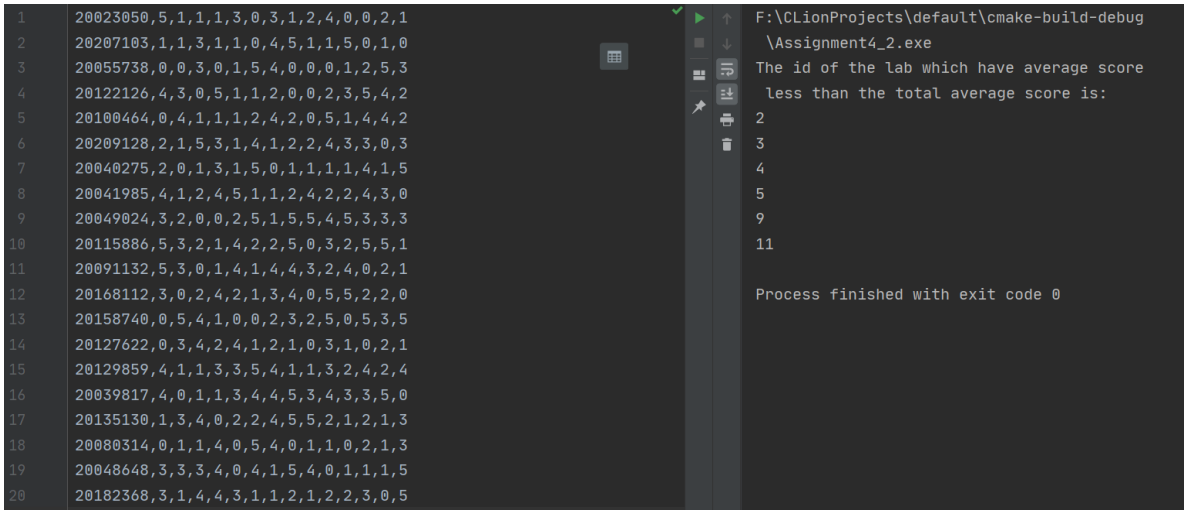




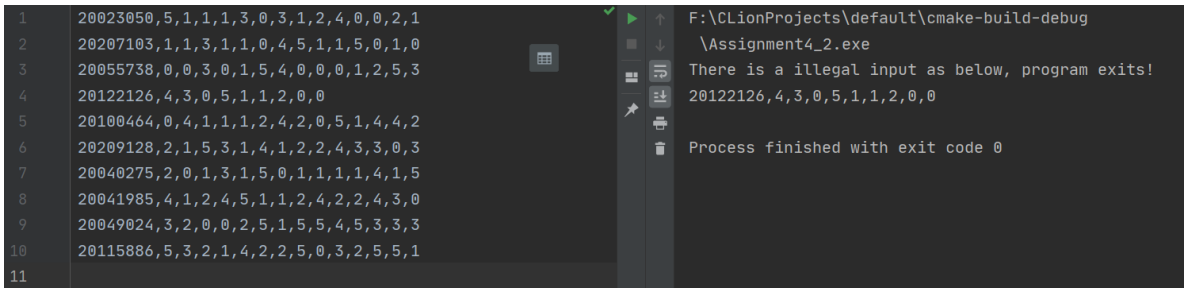
Q4

The data of the input file is shown at the left

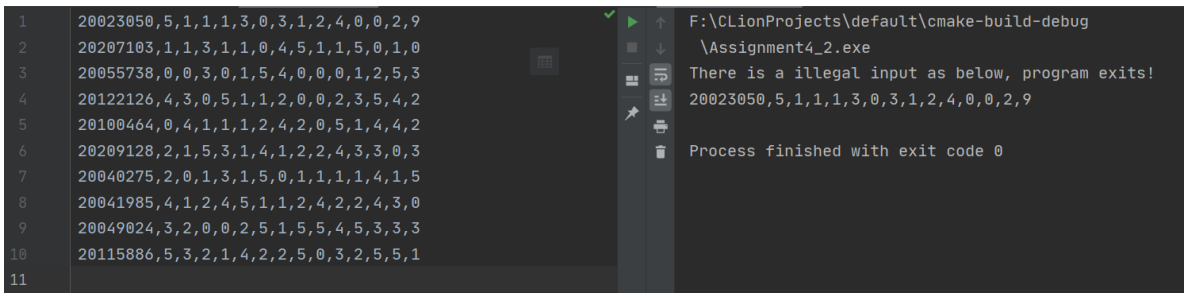




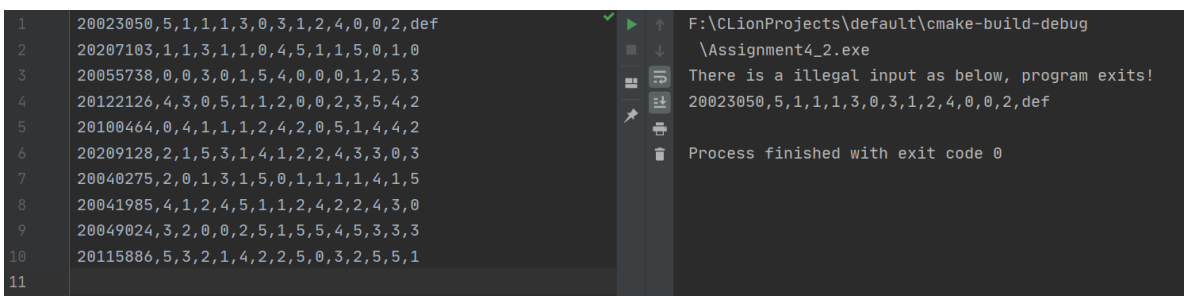
When one of the student's scores is not enough



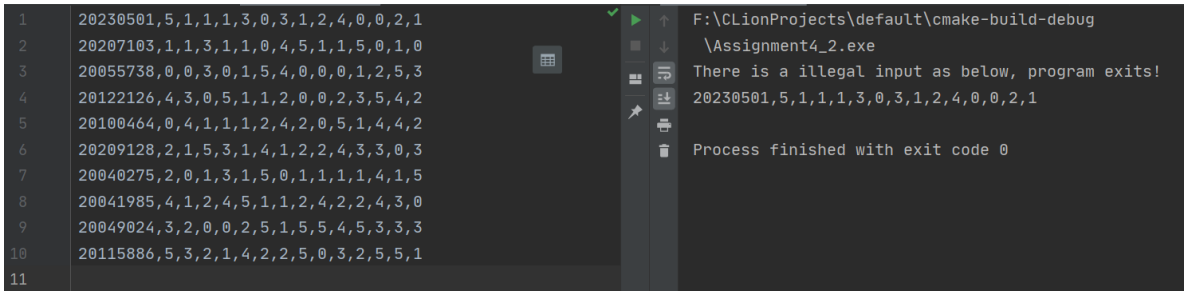
When a student has a lab score not between 0 and 5



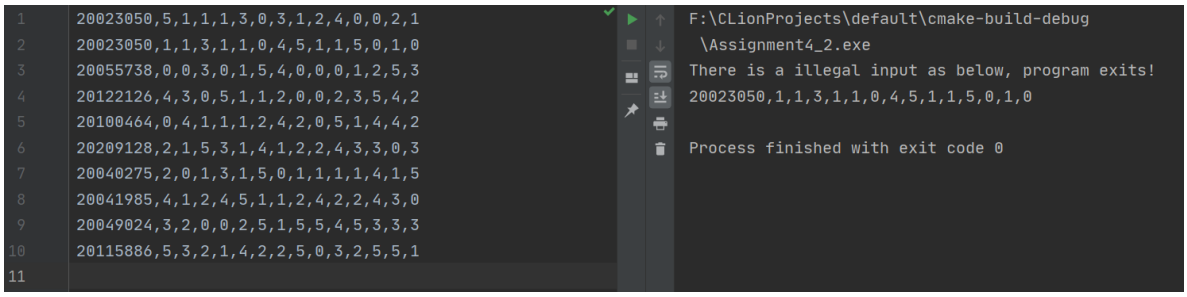
When the data has something wrong



When someone's id is illegal



When two students have the same id



Q5

The program is case-sensitive and will ignore the space at the both ends of the input

```
F:\CLionProjects\default\cmake-build-debug
 \Assignment4_3.exe
Program start!
hello world
Invalid command
start
command <start> recognized
stop
command <stop> recognized
restart
command <restart> recognized
reload
command <reload> recognized
status
command <status> recognized
goodbye
Invalid command
exit
Programme successfully exit!

Process finished with exit code 0
```

```
F:\CLionProjects\default\cmake-build-debug
 \Assignment4_3.exe
Program start!
 gogogo
Invalid command
  start
command <start> recognized
Status
Invalid command
status
command <status> recognized
EXIT
Invalid command
   exit
Programme successfully exit!

Process finished with exit code 0
```

Part 4 - Difficulties & Solutions

For q1, if we simply use the function `rand()`, we will always get the same result. A good solution is to provide different *seed* value used in a call to `srand()` and then the pseudo-random number generator can be expected to generate different results in the subsequent calls to `rand()`.