Challenge 1
Group 7

STUDENT ID	NAMES
20127369	Lê Quốc Trung
20127400	Phan Gia Huy
20127437	Dương Đức Anh
20127673	Trương Gia Huy

Working progess: 100%

Challenge 1
Group 7

Data structures:

```
struct Score {
    wstring ID;
    wstring name;
    int term;
    double grade;
    int credits = 0;
    int spec;
};
```

```
struct Student {
    wstring StudentID;
    wstring last;
    wstring first;
    wstring programID;
    vector<Score> score;
    vector<Score> foundation;
    double allGPA;
    double foundGPA;
    int Acc_Credits = 0;
};
```

Algorithms (step-by-step):

1. File and struct handling:
    - Read Grading file (Grading.csv) into struct<Student> GradeList (ReadStudent_Grading)

Challenge 1
Group 7

```
while (fin.tellg() < end)
{
    getline(fin,temp, ','); // Bo qua MAJ
    getline(fin, temp, ',');
    if (first_check == 0) temp2 = temp; // Lan dau tien set temp_ID2

    if (temp != temp2) {
        List.push_back(stu_temp);

        stu_temp.score.clear(); // Reset vector score trong Student

        first_check = 0;
    }

    temp2 = temp; // Gan gia tri ID de check cho den khi ID khac'

    stu_temp.StudentID = temp2;
    getline(fin, stu_temp.last, ',');
    getline(fin, stu_temp.first, ',');
    getline(fin, stu_temp.programID, ',');

    string Num_Temp;
    getline(fin, Num_Temp, ','); // Bo qua 1 cot AY

    getline(fin, Num_Temp, ',');

    sco_temp.term = stoi(Num_Temp);
    getline(fin, sco_temp.ID, ',');

    Vietlanguage();
    getline(fin, sco_temp.name, ',');
    ASCIIlanguage();

    getline(fin, Num_Temp, ',');   // Bo qua 1 cot Class

    getline(fin, Num_Temp, ',');
    sco_temp.grade = Num_Temp == "NULL" ? 0 : stod(Num_Temp);

    getline(fin, Num_Temp, ',');    // GradeType la` o^ tro^'ng

    getline(fin, Num_Temp);
    sco_temp.credits = stoi(Num_Temp);

    sco_temp.spec = checkSpec(sco_temp, sco_list);

    stu_temp.score.push_back(sco_temp);

    first_check = 1;
}
```

- Sort Grading file by ID and delete duplicated students (void sort(vector<Student>& bar))
    - Convert StudentID into integer and merge sort

Challenge 1
Group 7

```
string b1 = left[j].StudentID;
string b2 = right[k].StudentID;
int c1 = stoi(b1.erase(0, 2));
int c2 = stoi(b2.erase(0, 2));
```

```
void HopNhatNhungHocSinhVoTinhBiTrung(vector<Student>& List_Student_Grading, int foundation) {
    int j = 0;
    vector<Student>::iterator it1;
    vector<Student>::iterator it2;
    for (int i = 0; i < List_Student_Grading.size(); i++) {
        it1 = List_Student_Grading.begin() + i;
        int j = i + 1;

        int erase = 0;

        for (; j < List_Student_Grading.size(); j++) {
            if (List_Student_Grading[i].StudentID == List_Student_Grading[j].StudentID) {
                for (int z = 0; z < List_Student_Grading[j].score.size(); z++) {
                    List_Student_Grading[i].score.push_back(List_Student_Grading[j].score[z]);
                }
                it2 = List_Student_Grading.begin() + j;
                erase++;

                if (List_Student_Grading[i].StudentID != List_Student_Grading[j + 1].StudentID) break;
            }
            else break;
        }
        List_Student_Grading[i].foundGPA = CalcFoundGPA(List_Student_Grading[i], foundation);
        List_Student_Grading[i].allGPA = CalcAllGPA(List_Student_Grading[i]);

        if (erase > 0) {
            List_Student_Grading.erase(it1+1, it2+1);
        }
    }

}
```

- Read Interest file (Interests.csv) (ReadStudent_Interest)

Challenge 1
Group 7

```cpp
vector<Student> ReadStudent_Interest(string path) {
    vector<Student> List;

    fstream fin(path, fstream::in);

    fin.seekg(-1, std::ios_base::end);

    int end = fin.tellg(); // Ki tu cuoi cung cua .csv
    fin.seekg(0, std::ios_base::beg);

    fin.ignore(100, wchar_t(0xfeff)); // Bo ki tu dau tien

    fin.ignore(256, '\n'); // Bo line dau tien

    string temp;

    Student Stu_Temp;

    int debug = 0;


    for(int i = 0; i < 440; i++) {

        Stu_Temp.interest.clear();

        getline(fin, Stu_Temp.StudentID, ',');

        getline(fin, Stu_Temp.last, ',');

        getline(fin, Stu_Temp.first, ',');

        getline(fin, temp, ','); // REGDATE


        getline(fin, temp, ',');
        Stu_Temp.interest.push_back(temp);
        getline(fin, temp, ',');
        Stu_Temp.interest.push_back(temp);
        getline(fin, temp, ',');
        Stu_Temp.interest.push_back(temp);
        getline(fin, temp, ',');
        Stu_Temp.interest.push_back(temp);
        getline(fin, temp, ',');
        Stu_Temp.interest.push_back(temp);
        getline(fin, temp);
        Stu_Temp.interest.push_back(temp);

        List.push_back(Stu_Temp);
    }
    return List;
}
```

Challenge 1
Group 7

- Sort Interests file by ID (void sort(vector<Student>& bar))
- Add interests into GradeList

```
void HopNhatHocSinh(vector<Student> Interest_List, vector<Student>& Grade_List) {
    int j = 0;
    for (int i = 0; i < Grade_List.size(); i++) {
        for (j = 0; j < Interest_List.size(); j++) {
            if (Interest_List[j].StudentID == Grade_List[i].StudentID) {
                Grade_List[i].interest = Interest_List[j].interest;
            }
        }
    }
}
```

- Delete student that don’t have interests

```
void XoaNhungHocSinhKhongCoThamVong(vector<Student>& List) {
    auto it = List.begin();

    while (it != List.end()) {
        if ((*it).interest.size() < 6) {
            it = List.erase(it);
        }
        else it++;
    }
}
```

- Calculate GPAFound, GPA_All and credits

```
double CalcFoundGPA(Student a, int x) { // x: the number of all foundation courses
    if (countFound(a) < x)
        return 0;
    double temp = 0, credits = 0;
    for (int i = 0; i < a.score.size(); i++) {
        if (a.score[i].spec == 1) {
            temp += a.score[i].grade * a.score[i].credits;
            credits += a.score[i].credits;
        }
    }
    double fGPA = temp / credits;
    return roundDouble(fGPA);
}
```

Challenge 1
Group 7

```
double CalcAllGPA(Student a) {
    double temp = 0, credits = 0;
    int size = a.score.size();
    for (int i = 0; i < size; i++) {
        if (a.score[i].spec != -1) {
            temp += a.score[i].grade * a.score[i].credits;
            credits += a.score[i].credits;
        }
    }
    double AllGPA = temp / credits;
    return roundDouble(AllGPA);
}
```

```
int CalcAcc_NCredits(Student& a) {
    int credits = 0, size = a.score.size();
    for (int i = 0; i < size; i++) {
        if (a.score[i].grade >= 5)
            credits += a.score[i].credits;
    }
    return credits;
}
```

- Sort GradeList by GPAFound

```
void sort_grade(vector<Student>& bar) {
    if (bar.size() <= 1) return;

    int mid = bar.size() / 2;
    vector<Student> left;
    vector<Student> right;

    for (size_t j = 0; j < mid; j++)
        left.push_back(bar[j]);
    for (size_t j = 0; j < (bar.size()) - mid; j++)
        right.push_back(bar[mid + j]);

    sort_grade(left);
    sort_grade(right);
    mergeSort_grade(left, right, bar);
}
```

- Add students to Majors: Loop through the GradeList and in descending order for each student, we iterate the interests, if the major's quota is greater than 0 then we set the flag Chosen to student's interest index and Selected to the shorten form of the major.

Challenge 1
Group 7

```
void SapDatThamVong(vector<Student>&StuList, mapWSI Majors) {
    for (int i = 0; i < StuList.size(); i++) {
        if (StuList[i].Chosen != -1) continue;

        for (int j = 0; j < 6; j++) {

            if (Majors[StuList[i].interest[j]] > 0) {
                Majors[StuList[i].interest[j]]--;
                StuList[i].Chosen = j + 1;
                StuList[i].Selected = StuList[i].interest[j];
                break;
            }

        }
    }
}
```

2. Command Line and Arguments handling:
    - Command 1: Output Result.csv

```
void NhungNguoiXungDang(vector<Student> StuList, string path) {
    ofstream fout(path);
    fout << "StudentID,LastName,FirstName,RegDate,I1,I2,I3,I4,I5,I6,Selected,Chosen,GPA_Foundation,GPA_All" << endl;
    for (int i = 0; i < StuList.size(); i++) {
        fout << StuList[i].StudentID << "," << StuList[i].last << "," << StuList[i].first << "," << StuList[i].RegDate << ",";
        for (int iuthich = 0; iuthich < 6; iuthich++)
            fout << StuList[i].interest[iuthich] << ",";
        fout << StuList[i].Selected << "," << StuList[i].Chosen << "," << StuList[i].foundGPA << "," << StuList[i].allGPA << endl;
    }
}
```

    - Command 2: Search into the GradeList sequentially by ID and get the student's information

```
void TimNguoiThan(vector<Student> StuList, string TenNguoiThan) {
    for (int i = 0; i < StuList.size(); i++) {
        //TIM KIEM TUAN LOC
        if (StuList[i].StudentID == TenNguoiThan) {
            cout << StuList[i].StudentID << " " << StuList[i].last << " " << StuList[i].first << " " << StuList[i].RegDate << " ";
            for (int damme = 0; damme < 6; damme++)
                cout << StuList[i].interest[damme] << " ";
            cout << StuList[i].Selected << " " << StuList[i].Chosen << " " << StuList[i].foundGPA << " " << StuList[i].allGPA << endl;
        }
    }
}
```

    - Command 3: Loop through the GradeList, compare each student's selected which the given major and output it into Major.csv

Challenge 1
Group 7

```cpp
void writeMajor(vector<Student> major_list, string path) {
    ofstream fout(path + ".csv");

    // line dau tien
    fout << "StudentID,LastName,FirstName,RegDate,Chosen,GPA_Foundation,GPA_All" << endl;

    // may thang sinh vien
    for (int i = 0; i < major_list.size(); i++) {
        if (major_list[i].Selected == path) {
            fout << major_list[i].StudentID << ",";
            fout << major_list[i].last << ",";
            fout << major_list[i].first << ",";
            fout << major_list[i].RegDate << ",";
            fout << major_list[i].Chosen << ",";
            fout << major_list[i].foundGPA << ",";
            fout << major_list[i].allGPA << endl;
        }
    }

}
```

Reference: https://codereview.stackexchange.com/questions/167680/merge-sort-implementation-with-vectors