PROJETO 1 – AED

Schedule Manager

Class definitions

```
class Student {
public:
    Student();
    Student(string name, string studentCode, Schedule schedule);
    string get_name() const;
    Schedule get_schedule() const;
    string get_studentCode() const;
    set<string> get_belong_class() const;
    set<string> get_belong_ucs();
    void set_schedule(Schedule schedule);
    void set_name(string name);
    void set_studentCode(string studentCode);
    bool operator<(Student student) const;
    bool operator==(Student student) const ;
private:
    string name_;
    string studentCode_;
    Schedule schedule_;
};
```

```
class Lesson {
private:
    string uc_code;
    string weekday;
    float start_time;
    float duration;
    string type;
    string class_code;
public:
    Lesson() {};
    Lesson(string uccode, string weekday_, float starttime, float duration, string type, string class_code) ;
    string get_uc_code() const;
    string get_weekday() const;
    float get_start_time() const;
    float get_duration() const;
    string get_type() const;
    string get_class_code() const;
    void set_uc_code(string uccode);
    void setweekday(string weekday);
    void set_start_time(float starttime);
    void setduration(float duration);
    void settype(string type);
    void set_class_code(string class_code);
    bool operator==(const Lesson& other) const;
    bool operator<(const Lesson& other) const;
};
```

```
class Schedule {
public:
    Schedule();
    void set_lessons(vector<Lesson> lessons);
    void add_lesson(Lesson lesson);
    void remove_lesson(Lesson lesson);
    vector<Lesson> get_lessons() const;
private:
    vector<Lesson> lessons_;
};
```

```
class UcClass {
public:
    UcClass(string classCode, Schedule schedule);
    string get_classCode() const;
    Schedule get_schedule() const;
    void set_classCode(string classCode);
    void set_schedule(Schedule schedule);
    bool operator<(UcClass uclass) const;
private:
    string classCode_;
    Schedule schedule_;
};
```

Data processing and organization

```
#include <string>
#include <set>
#include <vector>
#include <fstream>
#include <sstream>
#include "UcClass.h"
#include "Student.h"
#include <iomanip>
using namespace  std;
set<UcClass> parsing_classes() ;
set<UcClass> parsing_schedules(set<UcClass> classes);
set<Student> parsing_students(set<UcClass> classes);
void print_schedule(Schedule schedule);
```

```
set<UcClass> parsing_classes(){
    ifstream in;
    in.open( s: "../Read_Info/classes_per_uc.csv");
    if (!in.is_open()){
        cout << "File not found!"<<'\n';
    }
    set<UcClass> classes;
    string skip;
    getline( &: in,  &: skip);
    while(in){
        string line;
        while(getline( &: in, &: line)){
            stringstream read( s: line);
            string UcCode,ClassCode;
            getline( &: read, &: UcCode, dlm: ',');
            getline( &: read, &: ClassCode, dlm: '\r');
            Schedule schedule;
            UcClass uclass(ClassCode, schedule);
            classes.insert( v: uclass);
        }
    }
    in.close();
    return classes;
}
```

```
bool UcClass::operator<(UcClass uclass) const{
    return classCode_ < uclass.get_classCode();
}
```

```
bool Lesson::operator<(const Lesson &other) const {
    map<std::string, int> weekday_to_int = {{ u1: "Monday",  u2: 1}, { u1: "Tuesday",  u2: 2}, { u1: "Wednesday",  u2: 3}, { u1: "Thursday",  u2: 4},
                                            { u1: "Friday",  u2: 5}, { u1: "Saturday",  u2: 6}, { u1: "Sunday",  u2: 7}};
    int this_weekday = weekday_to_int[this->weekday];
    int other_weekday = weekday_to_int[other.get_weekday()];

    // Compare by weekday first
    if (this_weekday < other_weekday) {
        return true;
    } else if (this_weekday > other_weekday) {
        return false;
    }

    // If weekdays are equal, compare by start_time
    return this->start_time < other.get_start_time();
}
```

Menu and Requests

```
class Schedule_Manager {
public:
    Schedule_Manager();
    void print_menu();
    void consult();
    void consult_schedule();
    void consult_students();
    void number_students();
    void consult_occuption();
    void uc_most_students();
    void add_uc();
    void remove_uc();
    void switch_students();
    void request();
    void print_history();
    void save_requests();
    void load_request();
```

```
int main() {
    Schedule_Manager schedulemanager;
    schedulemanager.load_request();
    while(true){
        schedulemanager.print_menu();
        char i;
        cin >> i;
        bool close = false;
        switch(i){
```

```
private:
    set<UcClass> classes;
    set<Student> students;
    queue <string> changes;
    map<pair<string,string>,vector<string>> get_uc_class();
    void upper(string& word);
    int get_min_attendance(string UcCode);
    bool check_student(string StudentCode);
    bool check_class(string ClassCode);
    bool check_uc(string UcCode);
    void automatic_remove(string StudentCode ,string UcCode,string ClassCode);
    void automatic_add(string StudentCode ,string UcCode,string ClassCode);
    void automatic_switch(string StudentCode ,string from_ClassCode,string from_UcCode,string to_ClassCode,string to_UcCode);
};
```

```
void Schedule_Manager::consult() {
    cout << "--------------------------Consult Menu--------------------------"<< '\n';
    cout << "| 1-Consult the schedule of a given student or class              |"<<'\n';
    cout << "| 2-Consult the students within a given class, course or year     |"<< '\n';
    cout << "| 3-Consult the number of students registered in at least n UCs   |" << '\n';
    cout << "| 4-Consult the class/year/UC occupation                          |" << '\n';
    cout << "| 5-Consult the UCs with the greatest number of students          |" << '\n';
    cout << "| 6-Menu                                                          |" << '\n';
    cout << "-------------------------------------------------------------------"<< '\n';
    cout << "Choose the option you want to do: ";
}
void Schedule_Manager::consult_schedule() {
    while (true){
        cout << "---------------Choose one---------------"<<'\n';
        cout << "|             1-Student               |"<<'\n';
        cout << "|             2-Class                 |"<<'\n';
        cout << "|             3-Back                  |"<<'\n';
        cout << "----------------------------------------"<<'\n';
        cout << "Choose the option you want to do: ";
        char i;
        cin >> i;
        bool cloose = false;
        if(i== '1'){
```

History and Changes

```cpp
void Schedule_Manager::print_history() {
    vector<string> historic;
    while(!(changes.empty())){
        string change = changes.front();
        historic.push_back(change);
        changes.pop();
    }
    bool cloose = false;
    while (true){
        cout << "-----------------------------------------Changes-----------------------------------------" <<'\n';
        int i = 1;
        for(auto change :string :historic){
            cout << "|"<<i<<"-"<<change<<'|'<<'\n';
            i++;
        }
        cout << "-----------------------------------------------------------------------------------------------"<<'\n';
        cout << "Do you want to undo the last change?(Y/N) ";
        char k;
        cin >> k;
```

```cpp
void Schedule_Manager::load_request() {
    ifstream in;
    in.open( s: "../Read_Info/requests.csv");
    if (!in.is_open()){
        cout << "File not found!"<<'\n';
    }
    while(in){
        string line;
        while(getline( &: in, &: line)){
            changes.push( v: line);
            stringstream in( s: line);
            vector <string> Code;
            string word;
            while(in >> word){
                Code.push_back(word);
            }
            if(Code[0]=="Added")automatic_add( StudentCode: Code[2], UcCode: Code[8].substr( pos: 0, n: Code[8].size()-1), ClassCode
            else if(Code[0]=="Removed")automatic_remove( StudentCode: Code[2], UcCode: Code[8].substr( pos: 0, n: Code[8].size()-1)
            else automatic_switch( StudentCode: Code[2], from_ClassCode: Code[11], from_UcCode: Code[14].substr( pos: 0, n: Code[14].siz
        }
    }
}
```

```cpp
void Schedule_Manager::save_requests() {
    ofstream out;
    out.open( s: "../Read_Info/requests.csv");
    if (!out.is_open()){
        cout << "File not found!"<<'\n';
    }
    while(!(changes.empty())){
        out << changes.front()<< '\n';
        changes.pop();
    }
}
```