PROJECT- AED

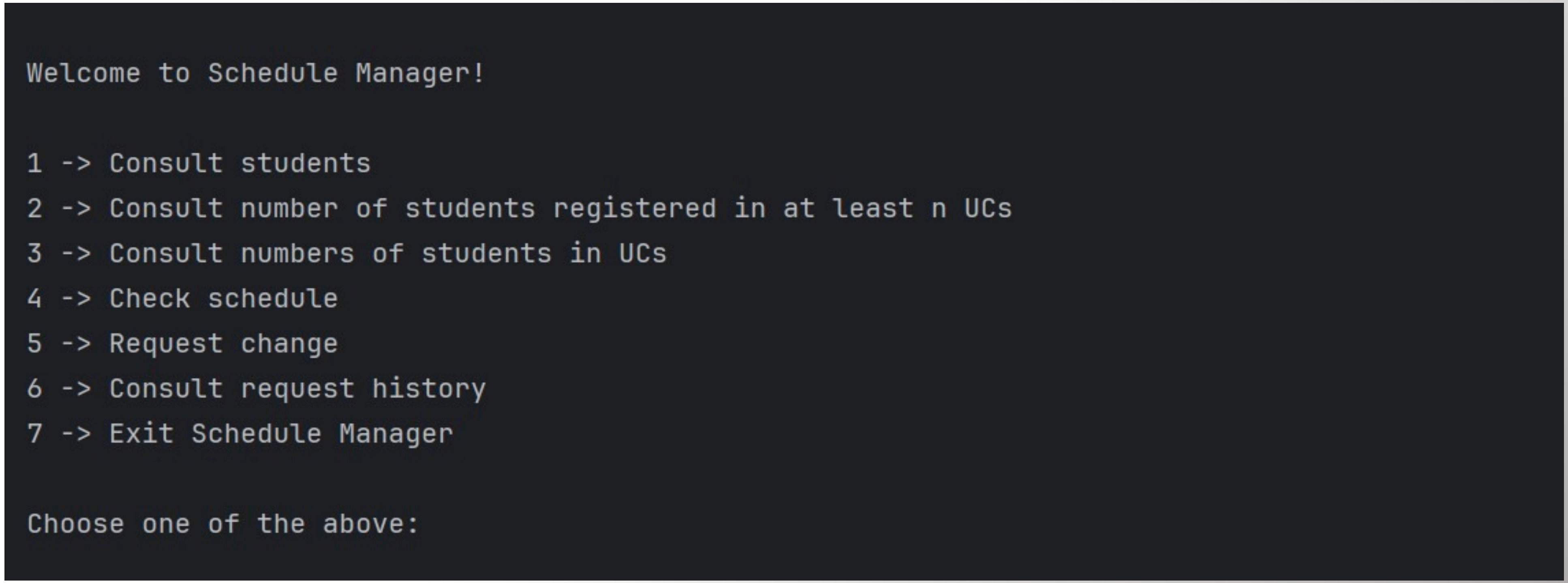
Schedule Management System

DUARTE MARQUES - 202204973

MARIA VIEIRA - 202204802

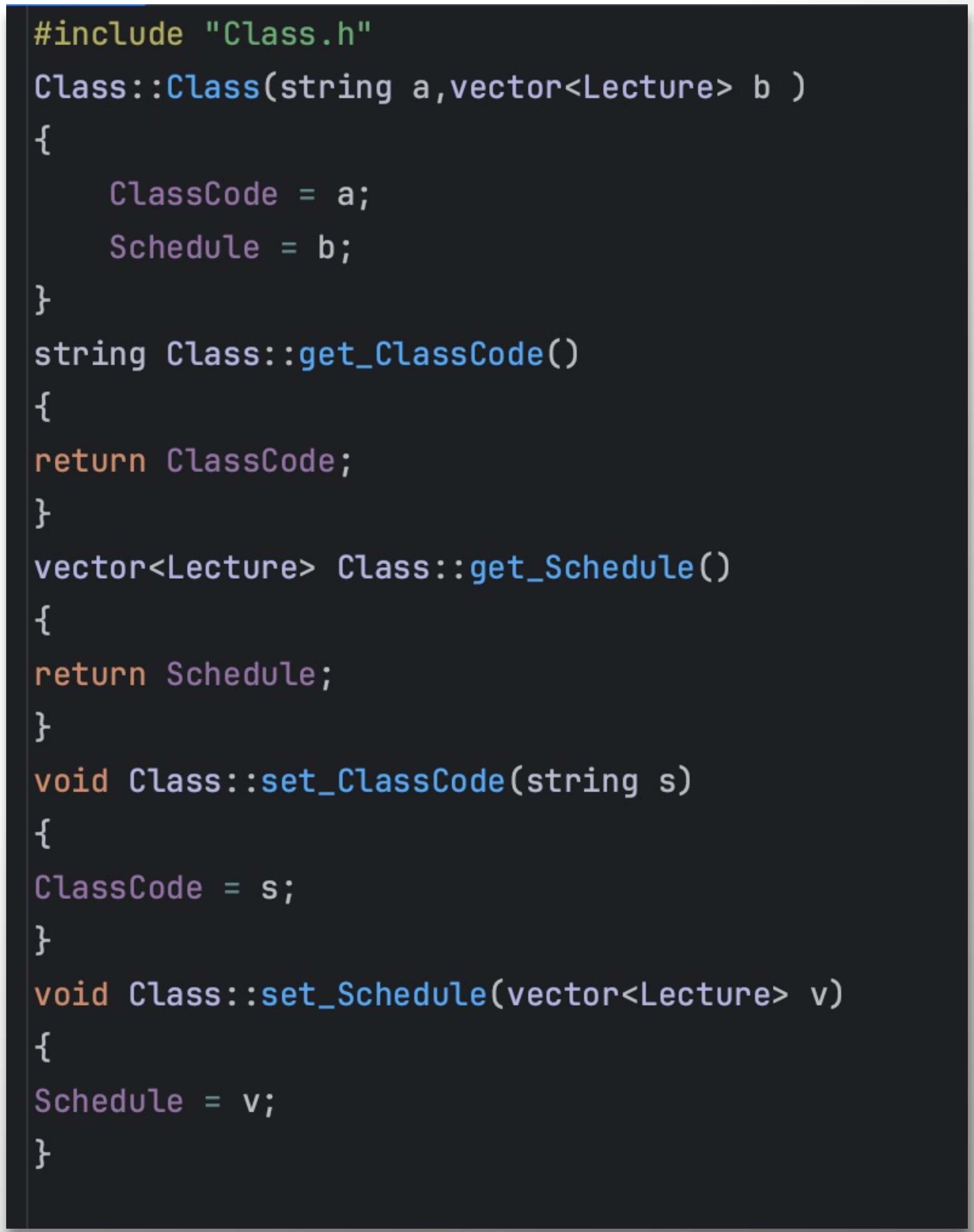
MARTA CRUZ - 202205028

INITIAL MENU



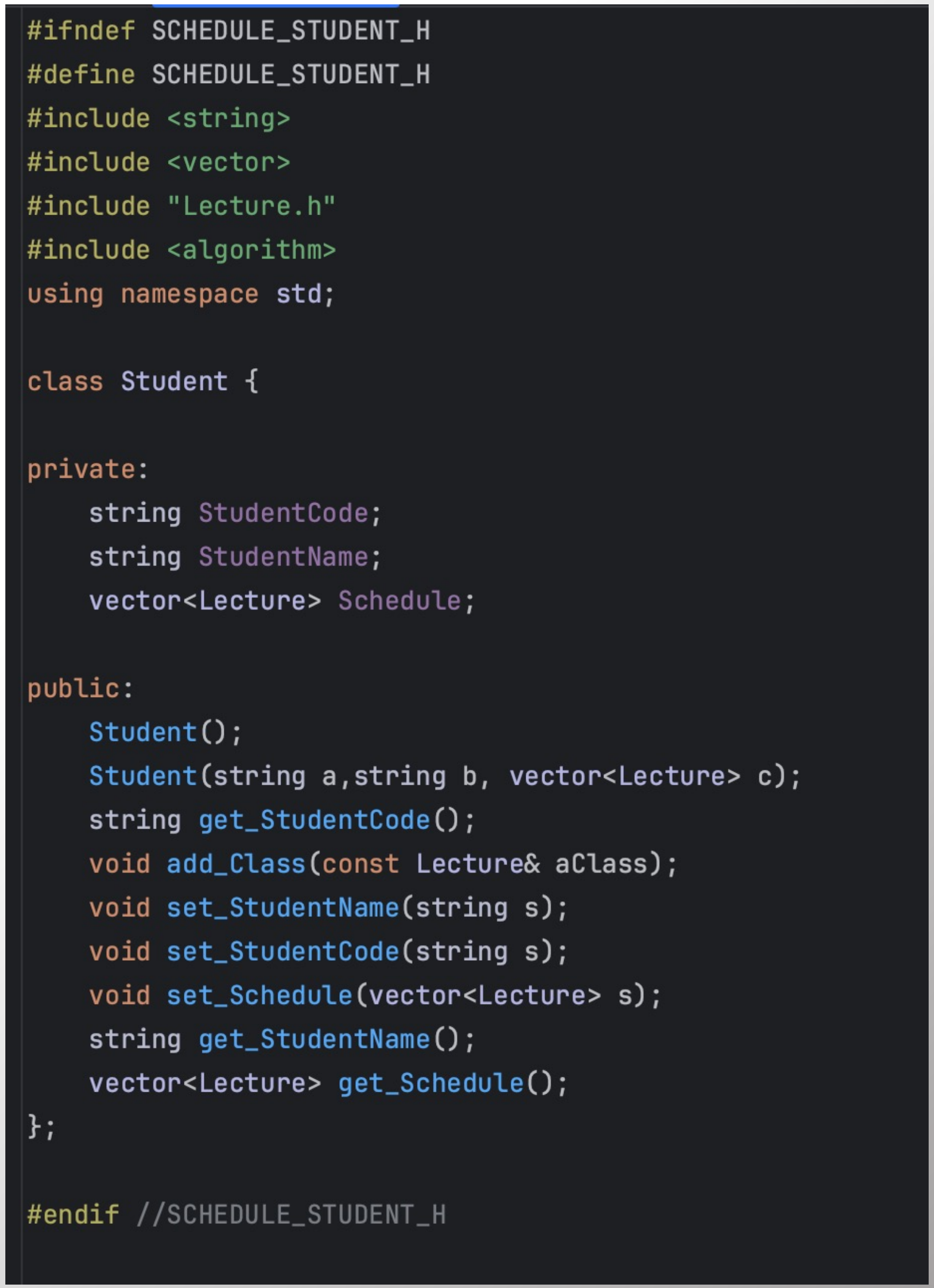
```
Welcome to Schedule Manager!

1 -> Consult students
2 -> Consult number of students registered in at least n UCs
3 -> Consult numbers of students in UCs
4 -> Check schedule
5 -> Request change
6 -> Consult request history
7 -> Exit Schedule Manager

Choose one of the above:
```

CLASS



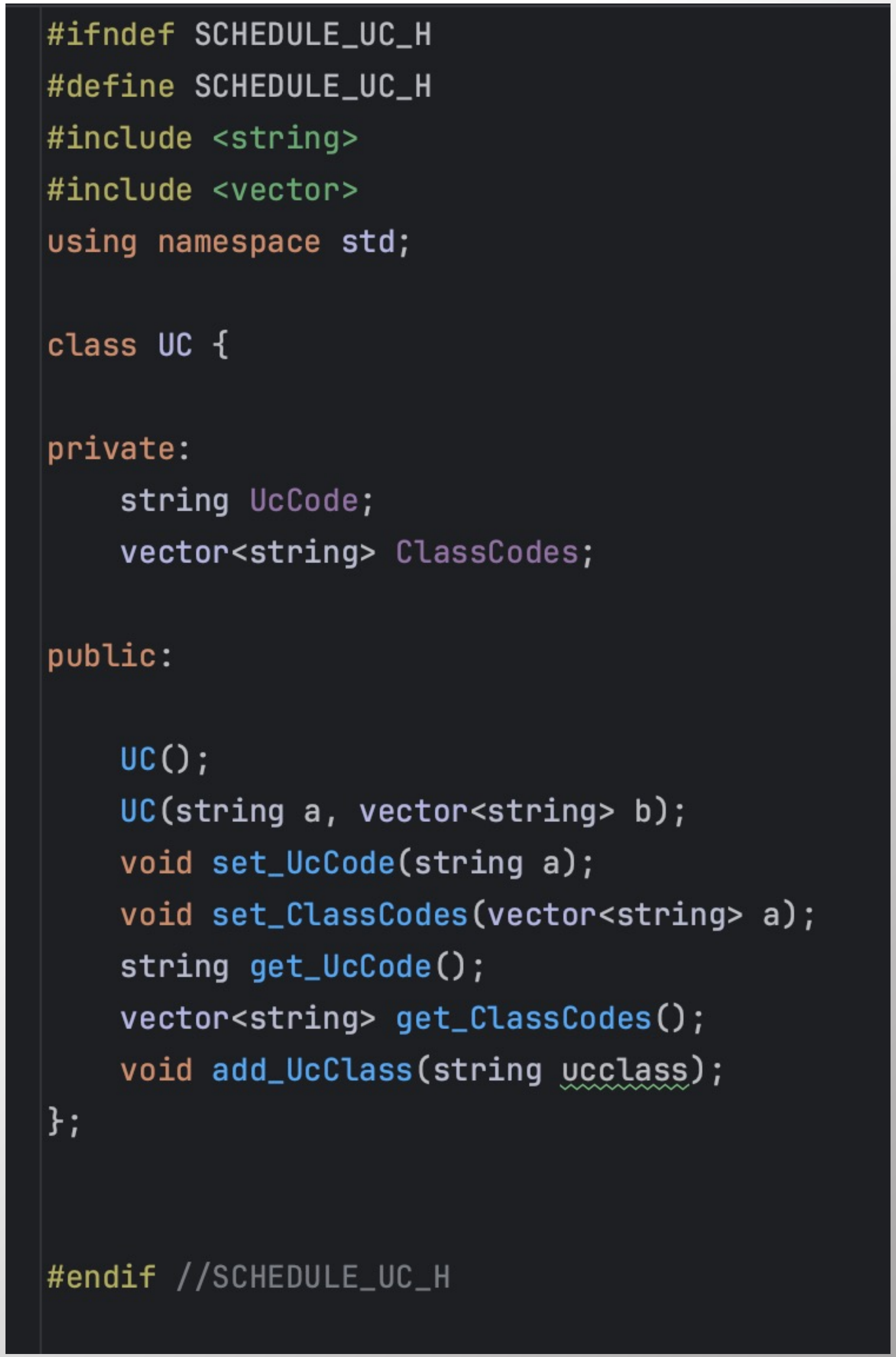
```
#include "Class.h"
Class::Class(string a,vector<Lecture> b )
{
    ClassCode = a;
    Schedule = b;
}
string Class::get_ClassCode()
{
return ClassCode;
}
vector<Lecture> Class::get_Schedule()
{
return Schedule;
}
void Class::set_ClassCode(string s)
{
ClassCode = s;
}
void Class::set_Schedule(vector<Lecture> v)
{
Schedule = v;
}
```

STUDENT



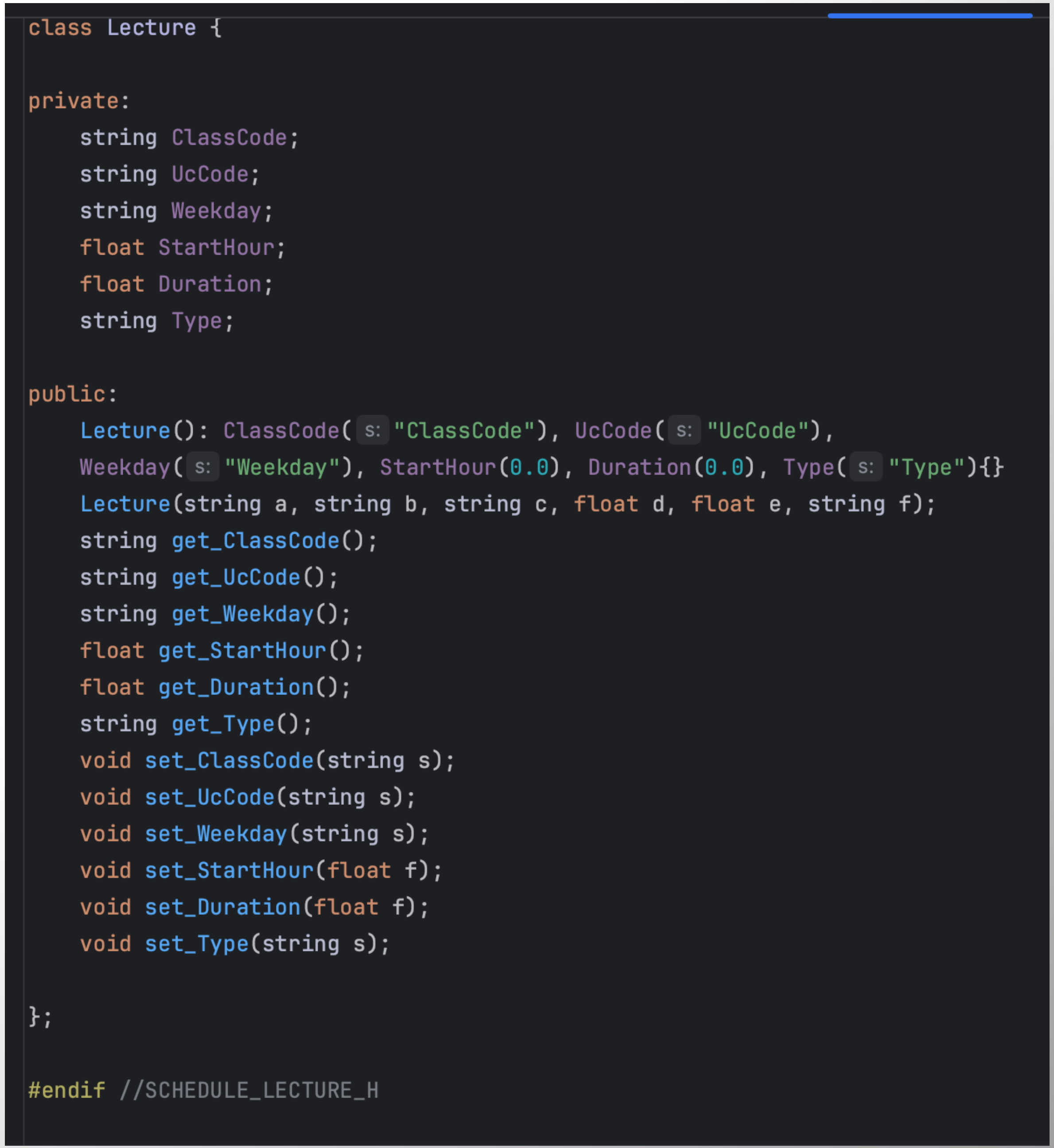
```cpp
#ifndef SCHEDULE_STUDENT_H
#define SCHEDULE_STUDENT_H
#include <string>
#include <vector>
#include "Lecture.h"
#include <algorithm>
using namespace std;

class Student {

private:
    string StudentCode;
    string StudentName;
    vector<Lecture> Schedule;

public:
    Student();
    Student(string a,string b, vector<Lecture> c);
    string get_StudentCode();
    void add_Class(const Lecture& aClass);
    void set_StudentName(string s);
    void set_StudentCode(string s);
    void set_Schedule(vector<Lecture> s);
    string get_StudentName();
    vector<Lecture> get_Schedule();
};

#endif //SCHEDULE_STUDENT_H
```

UC



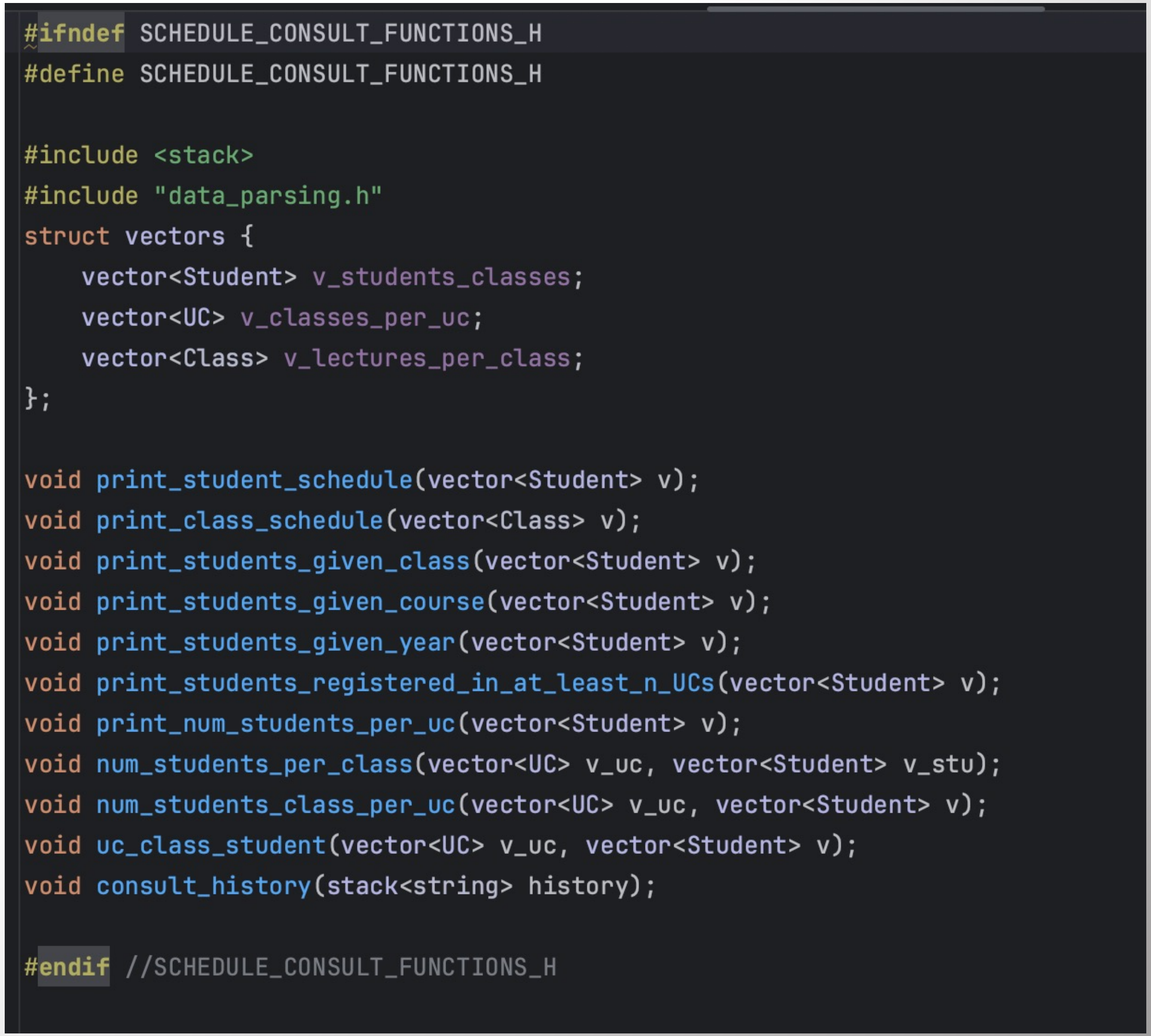
```cpp
#ifndef SCHEDULE_UC_H
#define SCHEDULE_UC_H
#include <string>
#include <vector>
using namespace std;

class UC {

private:
    string UcCode;
    vector<string> ClassCodes;

public:

    UC();
    UC(string a, vector<string> b);
    void set_UcCode(string a);
    void set_ClassCodes(vector<string> a);
    string get_UcCode();
    vector<string> get_ClassCodes();
    void add_UcClass(string ucclass);
};


#endif //SCHEDULE_UC_H
```

LECTURE



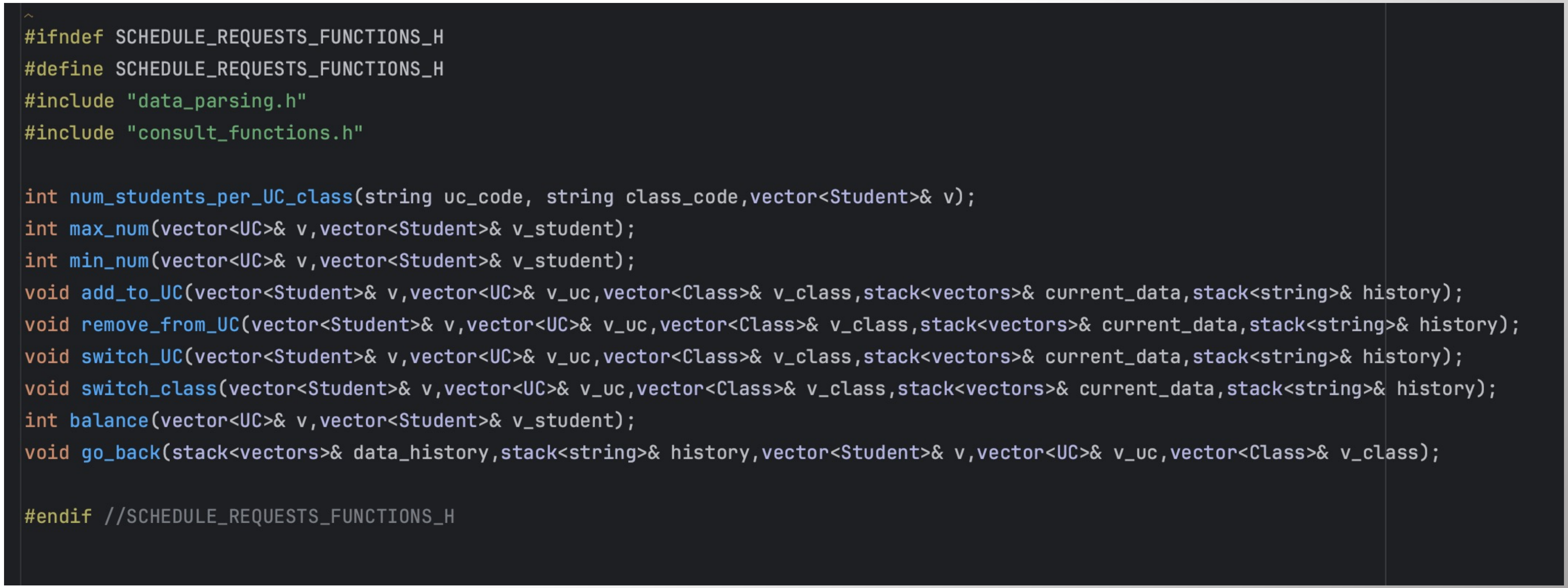
```
class Lecture {

private:
    string ClassCode;
    string UcCode;
    string Weekday;
    float StartHour;
    float Duration;
    string Type;

public:
    Lecture(): ClassCode( s: "ClassCode"), UcCode( s: "UcCode"),
    Weekday( s: "Weekday"), StartHour(0.0), Duration(0.0), Type( s: "Type"){}
    Lecture(string a, string b, string c, float d, float e, string f);
    string get_ClassCode();
    string get_UcCode();
    string get_Weekday();
    float get_StartHour();
    float get_Duration();
    string get_Type();
    void set_ClassCode(string s);
    void set_UcCode(string s);
    void set_Weekday(string s);
    void set_StartHour(float f);
    void set_Duration(float f);
    void set_Type(string s);

};

#endif //SCHEDULE_LECTURE_H
```

CONSULT FUNCTIONS



```cpp
#ifndef SCHEDULE_CONSULT_FUNCTIONS_H
#define SCHEDULE_CONSULT_FUNCTIONS_H

#include <stack>
#include "data_parsing.h"
struct vectors {
    vector<Student> v_students_classes;
    vector<UC> v_classes_per_uc;
    vector<Class> v_lectures_per_class;
};

void print_student_schedule(vector<Student> v);
void print_class_schedule(vector<Class> v);
void print_students_given_class(vector<Student> v);
void print_students_given_course(vector<Student> v);
void print_students_given_year(vector<Student> v);
void print_students_registered_in_at_least_n_UCs(vector<Student> v);
void print_num_students_per_uc(vector<Student> v);
void num_students_per_class(vector<UC> v_uc, vector<Student> v_stu);
void num_students_class_per_uc(vector<UC> v_uc, vector<Student> v);
void uc_class_student(vector<UC> v_uc, vector<Student> v);
void consult_history(stack<string> history);

#endif //SCHEDULE_CONSULT_FUNCTIONS_H
```

REQUESTS FUNCTIONS



```
#ifndef SCHEDULE_REQUESTS_FUNCTIONS_H
#define SCHEDULE_REQUESTS_FUNCTIONS_H
#include "data_parsing.h"
#include "consult_functions.h"

int num_students_per_UC_class(string uc_code, string class_code,vector<Student>& v);
int max_num(vector<UC>& v,vector<Student>& v_student);
int min_num(vector<UC>& v,vector<Student>& v_student);
void add_to_UC(vector<Student>& v,vector<UC>& v_uc,vector<Class>& v_class,stack<vectors>& current_data,stack<string>& history);
void remove_from_UC(vector<Student>& v,vector<UC>& v_uc,vector<Class>& v_class,stack<vectors>& current_data,stack<string>& history);
void switch_UC(vector<Student>& v,vector<UC>& v_uc,vector<Class>& v_class,stack<vectors>& current_data,stack<string>& history);
void switch_class(vector<Student>& v,vector<UC>& v_uc,vector<Class>& v_class,stack<vectors>& current_data,stack<string>& history);
int balance(vector<UC>& v,vector<Student>& v_student);
void go_back(stack<vectors>& data_history,stack<string>& history,vector<Student>& v,vector<UC>& v_uc,vector<Class>& v_class);

#endif //SCHEDULE_REQUESTS_FUNCTIONS_H
```