CENG 303 Term Project

Group Members:
- BERAT YILDIZER - 235101103
- MUSTAFA IBRAHIM AKSU - 235101102
- YUNUS EMRE MAVI – 225101154

Any compiler supporting at least C++ 20 can be used. We can generate main code in termProject.cpp and you can use it just customizing input csv files. Please locate csv files at the same location with the project. If our code cannot find input files, it will not work properly. You just need to change input files to test our program without customizing header columns. We performed many tests and tried many input sets😊. Also, after compiling and running our code, you can check console screen to be sure that exam scheduling is generated or not. If generated, go to `exam_schedule_program.csv` to see examination scheduling times.

Also, you can run .exe but please unzip all files to the same folder and run .exe in the same location with other project files.

Brief Introduction

First of all, we would prefer to give a brief introduction about which methods we used to satisfy term project requirements before explaining our code. We implemented an undirected graph structure to show relationships between classes. If a class is taken by at least 2 different students or is given by the same professor, we add a relationship to these classes. To put it in another way, our classes are vertex in the graph and there is an undirected edge between two classes if a class is taken by at least 2 different students or is given by the same professor as said above.

We decided to use the adjacency matrix method to implement graph structure. After implementing graph structure and showing relationships between classes(nodes) via edges, we used Graph Coloring method. If two classes or vertex has no connection, we assigned same color these classes(vertex). It means that examination time for classes having the same color can be at the same time and we defined examination times following that. The last point is that we decided to use Welsh Powell Graph coloring algorithm. So, Welsh Powell algorithm is implemented in `graphColouring`.

As a last point, output the final schedule, including assigned exam times for each class, and any blocked hour.

Functions and Their Explanation

Key functions implemented in the project are explained below.

1. `void readClassList(const std::string& filename, std::vector<classLists>& classes)`

   This function is implemented to read all classes from input csv file and then store them in a vector.CSV file shall be four different columns called StudentID, Professor Name, CourseID and ExamDuration(in mins). We provided an example input file and this input file should be located in the same path with .exe or if our code will be compiled again, it should be located in the project folder. Also, csv file name should be same since code implementation is done considering its name.

2. `void readRoomCapacities(const std::string& filename, std::vector<roomList>& rooms)`

This function is implemented to read room information's and store them in a vector. We provided an example input file and this input file should be located in the same path with .exe or if our code will be compiled again, it should be located in the project folder. Also, csv file name should be same since code implementation is done considering its name.

3. `map<string, int> readClassesfromVector(vector<classLists>& vec)`

This function is implemented to assign an ID to each classes. These ID's are used to implement graph via adjacency matrix and graph coloring. For example, if a class is assigned 4 as ID, index 4 is used for representing this class in adjacency matrix. We stored class and their ID's as pair in a map container which it is a AVL tree implementation.

4. `myGraph generateGraphForClasses(map<string, int> mymap, vector<classLists>& vec)`

This function is implemented to generate a graph using adjacency matrix method. As we provided in readClassesfromVector function,each class has an ID and these ID's are used to determine which index are used by which classes. if a class is taken by at least 2 different students or is given by the same professor, we set relevant index according to ID's.

5. `map<int, colorList> graphColouring(myGraph& graph, map<string, int>& mymap)`

This function is implemented to perform graph coloring using Welsh Powell algorithm. After assigning a color to each classes, they are stored in a map with together their ID's. As we said, ID's are assigned to each clas by readClassesfromVector function.

6. `vector<times> createExamSchedule(myGraph& graph, vector<classLists>& classes, std::vector<roomList>& rooms, map<string, int>& mapForClassandID, map<int, colorList>& mapForClassIDandColor);`

This function is implemented to determine examination times for each classes. While performing that, it is using assigned ID's for classes, generated graph structure, assigned colors, roomID's and their capacities. Each examination time is stored in a vector as a times class and this times class has class name, examination day, start hour, start min, finish hour, finish min, rooms members.

7. `void bookRoom(const char* room, const char* day, int start_hour, int start_min, int duration)`

This function is implemented to book a room. Please give room value using RoomLists.csv file. If you assign different room name, it will not work. Also, it is true for the day. It should be assigned following.

```
static string day_of_array[]{ "monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday" };
```

For example, room could be "A101" as it is located in RoomLists.csv file and also day could be "Monday" according to day_of_array object.

Last point, if you assigned an time off hour for booking, it will not run.

8. `times calculatingExamTimesforClass(vector<vector<vector<int>>>& room_time_list, const std::vector<roomList>& rooms, pair<int, int> duration, int numberOfStudents, int* day, int *hour, string className,int* sayac);`

This function is implemented to be used by **`createExamSchedule`** function. This implementation may be done in **`createExamSchedule`** without generating this function but it will be complex for reader.

You can see the example output below.

----------MONDAY----------
9:0 - 10:15 CENG201 - Rooms: A104; A105;
9:0 - 9:40 CENG300 - Rooms: A108;
9:0 - 9:30 CENG403 - Rooms: A109; A110; A111; A112; A113; A114;
9:0 - 9:45 ENG101 - Rooms: A115; A116;
11:0 - 12:0 CENG101 - Rooms: A104; A105; A108;
11:0 - 11:45 CENG209 - Rooms: A109; A110; A111;
11:0 - 11:30 CENG301 - Rooms: A112;
11:0 - 12:0 CENG405 - Rooms: A113; A114; A115; A116; A117;
13:0 - 14:0 CENG113 - Rooms: A104; A105; A108;
13:0 - 13:50 CENG207 - Rooms: A109; A110; A111;
13:0 - 14:15 CENG303 - Rooms: A112;
13:0 - 14:15 CENG408 - Rooms: A113; A114; A115; A116; A117;
15:0 - 16:30 CENG307 - Rooms: A104; A105;
15:0 - 15:45 CENG407 - Rooms: A108; A109; A110; A111; A112; A113;
15:0 - 16:40 CHEM101 - Rooms: A114; A115; A116;
15:0 - 15:30 ENGR201 - Rooms: A117; A118;
----------TUESDAY----------
9:0 - 10:30 CENG203 - Rooms: A101; A102;
9:0 - 10:0 CENG305 - Rooms: A103;
9:0 - 11:20 CENG400 - Rooms: A104; A105; A106; A107; A108;
9:0 - 11:0 MATH101 - Rooms: A109; A110; A111;
12:0 - 13:20 CENG401 - Rooms: A101; A102; A103; A104; A105;
12:0 - 12:45 CENGXXX - Rooms: A106;
12:0 - 14:0 ENGR260 - Rooms: A107; A108;
12:0 - 14:15 PHYS101 - Rooms: A109; A110; A111;
15:0 - 17:15 PHYS103 - Rooms: A101; A102; A103;
----------WEDNESDAY----------
9:0 - 9:40 TDL101 - Rooms: A101; A102; A103;
10:0 - 10:30 TÃ°T101 - Rooms: A101; A102; A103;
----------THURSDAY----------
----------FRIDAY----------
----------SATURDAY----------
----------SUNDAY----------

-------bookedHours--------		
-------Day--------	----------hour----------	----------rooms----------
monday	from 9 : 30 to 18 : 30	A101
monday	from 9 : 0 to 18 : 0	A102
monday	from 9 : 0 to 18 : 0	A103
monday	from 9 : 0 to 18 : 0	A107
monday	from 9 : 30 to 18 : 30	A106