**Group** : Error404Fixed

**Problem** : Automatic time table generator (P1).

**GitHub link :**<https://github.com/Williamsonlegend81/Capstone_Challenge_PS-1_Error404Fixed.git>

**MEMBERS :**

1. Pala Aaditya Vimalkumar – 202301076

2. Vrund Hemang Kansara – 202301015

3. Meetkumar Deepakkumar Patel – 202301091

4. Panchal Yash Kalpeshkumar – 202301094

## **Constraints :**

1. Reading is done in csv files.

2. We have created six csv files and one csv file for reading room information:

· For Autumn : Autumn, autumnMasters, MnCEVDAutumn.

· For Winter : Winter, winterMasters, MnCEVDWinter.

3. Make sure that the in csv file for room, all the rooms must be sorted in increasing order of their capacity.

4. The output of the program is given in an excel file, which will automatically pop-up.

## **Limitations :**

1. From **semester 6** onwards, there are many electives and very few core subjects, so our code doesn’t differentiate between core and electives, although it gives correct output in most of the cases, but there might be **very few** cases where the core subject will collide with the electives.

2. Our code treats the ICT/CS and MnC/EVD students separately. Means it considers there will be **NO COMMON LECTURES** for ICT/CS and MnC/EVD students.

3. The code can handle max up-to 50-60 subjects in ICT/CS (considering electives) and 10-15 in MnC/EVD/Masters. For input size greater than this, there might be some lectures which will not get sufficient lectures.

## **Plus points :**

1. The **generateTimeTable** function is the core function of the program. It can be used to generate any time table, provided correct input order in csv files, considering sections and lecture hours, and also considering the limitations of the input.

2. There is no collision of classroom and instructor.

3. The time table gives slot-wise output, slot m1 for 8-8:50, and so on.

4. Code also provides time table for the instructor.

## **Data structures used :**

1. We have used vector to store information about course. Vector is dynamic in nature, it can grow or shrink its size, and we don’t know how many lectures we will be given, so to optimize space complexity, we have used vector.

2. Map to store room’s information and to store instructor time table. Map stores the information in key-value format. In our case, the key is the room name and its capacity (for room) and the name of the instructor (for instructor time table). Map is also dynamic in nature. It takes O(log n) time to store the value and O(n\*log n) time to give output because it stores the data in increasing order of the key.

## **Explanation of Code :**

Struct room :- User data type, string which stores the name of the room and capacity to store the capacity of the room.

Class day : Attributes of course in a day, course name, instructor name, room, program, section and semester.

Class course : Attributes specific to course like lecture hours, number of students, and is Repeated for the day. We have also created 5 vectors representing the days (Monday to Friday). This can give time table for a particular day.

Two functions which reads the course information and room information from csv file and stores it in a vector.

Struct instructor time table : attributes for instructor’s time table like room, time, day, course.

Function to generate instructor time table : It generates the instructor’s time table and stores it in the map.

**The core function of the program :- Generate time table :**

1. There are two vectors, one to store the time table for the day and other checks if the instructor is repeated in the same slot. This check helps to avoid collision of instructor in the same slot.

2. There are three nested loops.

· The first loop is the loop of semester (2,4,6,8 for winter and 1,3,5,7 for autumn).

· First it will check if the remaining lecture hours of the subject is not zero.

· The second loop is the loop for assigning rooms. It will first check if the capacity of the room is greater than or equal to the number of students in the course.

· First it will check if the instructor is repeated in the slot or not. The single function call creates the time table for one slot. We check if the instructor is repeated in the slot, if not then it will assign the lecture. If it is repeated, then it will go for another iteration to allocate other lecture.

· The third nested loop is for handling sections. It will check if the instructor is same or not. If the instructor is same for the same course, then it will allocate other course. If the instructor is not same, then it will assign the same lecture for section A or section B.

This process is repeated for all other batches (year 1, year 2, …, MnC year 1,…, Masters,…).

· In between the this loops, the values of remaining lecture hours, updating the room is occupied or not is updated.

· The instructor’s time table is also generated along with these loops.

· It is stored in the map, by calling the function generate instructor time table.

3. The program works in O(n­2) in almost all cases.

Print time table function : Prints the time table according to the day.

Print free slot time table : This function prints the free slot time table (the lectures which doesn’t fit in the morning shift).

Make slot false function : This function sets all the values of the room to false. False means room is unoccupied, true means room is occupied.

Main function (Driver function) :

Here, the function of reading csv files is called, and then segregating the readed file according to semesters.

It also checks if any lecture is remaining, if it is remaining, then it will be allocated in free slot.

## **The asymptotic analysis of the code.**

Let the size of the input be ‘m’. We are not considering the time taken to read the csv files.

For segregation, it will take almost 3\*m time. In our code, the maximum input is in the autumn or winter file, followed by MnCEVD and masters.

There are 5 week days and 7 time slots, therefore the function will be called 35 times.

Let there are ‘n’ subjects in each year and ‘r’ rooms. Let there are ‘i’ instructors in each slot. The unique instructor check will take O(i) time. However, since the number of unique instructors in each slot is will not be very large, we can approximate it to O(1) time.

Further, the allocation for section ‘A’ and section ‘B’ will also take the same time. Therefore, for ‘n’ subjects and ‘r’ rooms, the combined complexity will be n2 \* (r\*log(r))2.

This is done 17 times(year1,..year4,MnC y1, …, MnC y4, and other master’s branches).

Time to make false each slot and room will be O(n) and O(r).

The function is called 35 times, so,

Total time complexity = 35\*(17 \* n2 \* (r\*log(r))2) + n + r + 3\*m

= 595 \* n2 \* (r \* log(r))2) + n + r + 3\*m

In O terms, time complexity will be O(n2) (since n >> r and n2 >> m).

**Pseudocode for the generate time table function :**

for(iterate over all subjects in a year/branch) {

if remaining-lectures != 0 then {

for(iterate over all rooms) {

if capacity-of-room >= number\_of\_students && !room-is-occupied && !lecture-is-repeated then {

if instructor-not-found-in-same-slot then {

m.push(instructor\_Name,course\_Code,room\_name, semester,course\_name, section, programme)

instructor.push(instructor\_Name)

remaining-lectures = remaining-lecture - 1

room-is-occupied = true

lecture-assigned-in-the-slot++

lecture-is-repeated = true

if section = "A" OR section = "B" then {

for(iterate over all subjects) {

if remaining-lectures != 0 then {

lecture-allocated-in-other-section = false;

if sections-are-different && instructors-are-different then {

for(iterate over room) {

if capacity-of-room >= number\_of\_students && !room-is-occupied && !lecture-is-repeated then {

m.push(instructor\_Name,course\_Code,room\_name, semester,course\_name, section, programme)

instructor.push(instructor\_Name)

remaining-lectures = remaining-lecture - 1

room-is-occupied = true

lecture-is-repeated = true;

lecture-allocated-in-other-section = true;

generateInstructorTimeTable(instructor\_Name,course\_Code,room,day,time)

stop

}

}

}

if lecture-allocated-in-other-section then

stop

}

}

}

generateInstructorTimeTable(instructor\_Name,course\_Code,room,day,time)

stop

}

}

}

}

if lecture-assigned-in-the-slot then

stop

}

The output of the program is given in output.csv file, which will change every time when you run the program. Excel sheet will pop-up to show the output.

