CAPSTONE PROJECT
Subject : Data Structure (IT205)
B. Tech (ICT), 2nd Semester,
Group (Team) – Quadcoders
Project: Automated Time-Table Generator(P1)

Problem:
To generate a weekly lecture and lab timetable with no clashes and continuous sessions of the same instructor, an algorithm must assign lectures and labs for each course while considering instructor availability and course credit details. It should accommodate the possibility of multiple instructors for the same course and prioritize scheduling lectures during initial hours for certain instructors. Finally, the algorithm should allocate classrooms or labs based on availability and capacity, resulting in a finalized timetable meeting all constraints.

Members:
Bhavya Bhanvadiya(202301123)
Devansh Shah(202301136)
Shamit Gandhi(202301041)
Tirth Boghani(202301025)

Github Link To Repository:
https://github.com/Shamit2105/Timetable

Pseudocode:

```
Start:
  Declare necessary libraries and namespaces.

Define Class TTGen:
  This class encapsulates the functionality to generate timetables.

  Declare public member variables:
    These variables store information related to courses, faculty, classrooms, and timetable grid.

    Declare 2D vectors for:
      - Course IDs
      - Course names
      - Credits
      - Faculty
      - Number of hours
      Each vector represents data for different semesters and courses within each semester.

    Declare 1D vectors for:
      - Classrooms
      - Time slots
      - Days
      These vectors store information about classrooms, time slots, and days of the week.

    Declare a 3-dimensional vector to represent the timetable grid:
    The timetable grid stores the schedule of courses, classrooms, and faculty across time slots
      and days.

  Constructor:
    This method initializes the TTGen object and prompts the user for necessary inputs.

    Take input for semester type (winter or autumn) and classroom details.
    The semester type and classroom details are necessary for generating the timetable.

    Initialize the timetable grid.
    The timetable grid is initialized to represent the schedule for different semesters, days, and
      time slots.

  input():
    This method takes input from the user for course details.

    Resize vectors for each semester and course.
    The vectors storing course details are resized based on the number of courses in each
      semester.

    Print statements to guide user input.
    Instructions are provided to the user to input course details for each semester.

    Take input for course details.
    The user inputs course IDs, names, credits, and faculty for each course in each semester.

  generateTimetable():
    This method generates the timetable by assigning courses to time slots and days.

    Declare a 2D vector to track assigned lectures.
```

The lectures_assigned vector keeps track of the number of lectures assigned for each course.

Declare a map for faculty assigned and last faculty.
These maps keep track of faculty assigned to each time slot and day, and the last faculty who taught in a slot, respectively.

Iterate over time slots, semesters, and days:
Loop through each possible combination of time slot, semester, and day.

Ensure every faculty is initially available.
At the beginning of each time slot, faculty availability is reset.

Assign a classroom to each semester.
Each semester is assigned a specific classroom for conducting lectures.

Iterate through days and courses:
Loop through each day and course to assign lectures.

Assign lectures to faculty sequentially.
Courses are assigned to faculty members sequentially, ensuring no double-booking.

Check for faculty availability and breaks.
Faculty availability and the need for breaks are checked before assigning lectures.

Update timetable grid accordingly.
The timetable grid is updated with assigned courses, classrooms, and faculty.

displayFacultySchedule(faculty_name):
This method displays the timetable for a specific faculty member.

Create a file for faculty timetable.
A file is created to store the timetable for the faculty member.

Iterate through days and time slots:
Loop through each day and time slot to retrieve the faculty's schedule.

Print faculty's schedule systematically.
The faculty's schedule for each day and time slot is printed in a systematic format.

displayClassroomTimetable(classroom_name):
This method displays the timetable for a specific classroom.

Create a file for classroom timetable.
A file is created to store the timetable for the classroom.

Iterate through days and time slots:
Loop through each day and time slot to retrieve the classroom's schedule.

Print classroom's schedule systematically.
The classroom's schedule for each day and time slot is printed in a systematic format.

main():
-> Initially deleted all the textfiles generated as it was opened it in append mode in the functions.
-> Called the TTGen function and passed the input file through command prompt through command a.exe < textfile.txt
->Passed input for printing the other two timetables(in text file itself)

Time Complexities and Space Complexities:

Input Processing (input() void): 2-D vector was the most complicated data structure in our input so the time complexity and space complexity will depend upon it:
Time Complexity: As we enter all elements in the vector sequentially by iterating through it, the time complexity will be O(n*m) where n is number of semesters and m is maximum number of courses in a semester.

Timetable Generation (generateTimetable() void):
Space Complexity of Additional Data Structures:
i)faculty_assigned **(map<pair<int, int>, unordered_set<string>>):** The space complexity depends on the number of unique faculty names assigned to each time slot and day. Let's denote the average number of faculty members assigned per time slot and day as f_avg. Therefore, the space complexity of faculty_assigned can be expressed as O(n * d * f_avg), where n is the number of time slots, and d is the number of days.

ii)last faculty:(map<pair<int, int>, string>): The space complexity depends on the number of time slots and days for which the last assigned faculty member is tracked. Let's denote the average number of faculty names tracked per time slot and day as f_last. Therefore, the space complexity of last_faculty can be expressed as O(n * d * f_last).

iii)Temporary variables for scheduling operations: The space complexity of these variables depends on their usage within the algorithm and the size of the input data. Since they are temporary and typically not stored beyond the scope of function calls, their contribution to overall space complexity is relatively small and can be considered constant or negligible.

As we had to iterate through days, timeslots, semesters and courses of each semester sequentially, the time complexity and space complexity will be O(t * d * n * m), as O(t * d * n * m) is greater then space complexity of all three above mentioned space complexity, where t is the number of time slots, n is the number of semesters, d is the number of days, and m is the maximum number of courses per semester.

Output File Generation (writing timetable.txt): Same just like generateTimetable(), as we had to iterate through days, timeslots, semesters and courses of each semester sequentially, the time complexity and space complexity will be O(t * d * n * m), where t is the number of time slots, n is the number of semesters, d is the number of days, and m is the maximum number of courses per semester.

Display Faculty Schedule (displayFacultySchedule() method): Same, as we as we had to iterate through days, timeslots, semesters and courses of each semester sequentially, the time complexity and space complexity will be O(t * d * n * m).

Display Classroom Timetable (displayClassroomTimetable() method): Same, as we as we had to iterate through days, timeslots, semesters and courses of each semester sequentially, the time complexity and space complexity will be O(t * d * n * m).

Space Complexity of Global Variables and Constants: Since the provided code does not contain any significant global variables beyond the class definition and some constants, their space complexity can be considered negligible.

Therefore, the time complexity of our entire algorithm is O(t * d * n * m).

Therefore, the space complexity of our entire algorithm is O(t * d * n * m).

If we consider t*d as constant(as no. of days and no of time slots will always be same no matter changes in no. of semesters) then the time complexity and space complexity of our program and last four of the above mentioned functions can be considered as:
Time Complexity = O(n * m)
Space Complexity = O(n * m)

Why Choose This Data Structure?:

Vectors for Course Details (Course IDs, Names, Credits, Faculty, Number of Hours):

Rationale: We chose vectors for storing course details because they provide dynamic resizing, random access, and sequential storage. This was helpful for us in managing variable-sized data such as course information for different semesters. Dynamic resizing allowed us to accommodate varying no of courses flexibly in different semesters without preallocation. Random access allowed us to recall and manipulate course details quickly during timetable generation.

Vector for Classrooms, Time Slots, and Days:

Rationale: Vectors are selected for storing classrooms, time slots, and days because they provide sequential access and dynamic resizing, which are suitable for managing fixed-size lists of elements. Sequential access allowed us to easily iterate through classroom names, time slots, and days. This led to straightforward data retrieval and manipulation. Dynamic resizing accommodates changes in the number of classrooms, time slots, or days without fixed size constraints. Whenever 2-D vectors were used, they were generally used for storing data(column) of a particular semester(row) to that semester itself so that access was smooth and not confusing.

3-Dimensional Vector for Timetable Grid:

Rationale: A 3D vector is used to represent the timetable grid because it provides a structured way to organize course schedules across different time slots, days, and semesters. A tuple was used inside it to make insertion and retrieval of data smooth. By using a tuple, the timetable grid can neatly encapsulate all relevant details of a course schedule entry within a single element. This structured representation simplifies access to and manipulation of timetable data, as each tuple holds all necessary information for a specific time slot, day, and semester. The tuples offered a concise and expressive way to represent related data elements within the timetable grid, enhancing code readability and maintainability. Even though the time complexity became increasingly expensive, we decided to use this for avoiding any confusion while performing operations on it.

Map for Faculty Assigned:

Rationale: This map keeps track of faculty assigned to each time slot and day. The key is a pair of integers representing the time slot and day, while the value is an unordered set of strings containing the names of faculty members assigned to teach during that time slot and day. Using a map allows for efficient lookup based on time slot and day, which is the main requirement for managing faculty schedules. By using an unordered set for the faculty names, we ensured that each faculty member is assigned to a specific time slot and day only once, preventing overlapping assignments as set will never have a repeated element.

Map for Last Faculty:

Rationale: This map keeps track of the last faculty member assigned to each time slot and day. It aids in providing breaks to faculty members if needed. By storing the last faculty member assigned to each time slot and day, the algorithm can identify cases where a faculty member may require a break between consecutive classes. It enables the algorithm to consider faculty preferences or constraints regarding breaks between classes, ensuring a balanced workload and optimizing faculty schedules.

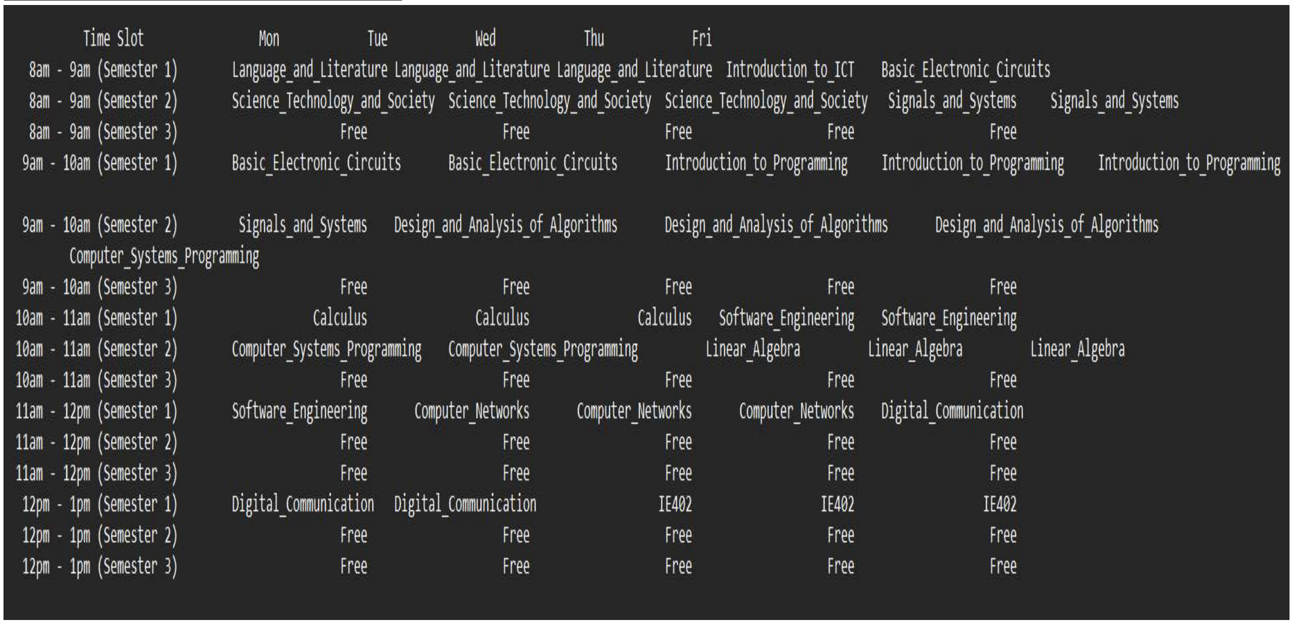
Why use map in both the above cases:

> **Efficiency:** Maps provide efficient lookup and insertion operations, which is important for managing faculty assignments and tracking last assigned faculty.
>
> **Association of Time Slots and Days:** Pairing time slots and days as keys allowed us to easily connect faculty assigning and last assigned faculty with specific time slots and days.
>
> **Flexibility:** Using unordered sets within faculty_assigned allowed us to prevent multiple faculty members to the same time slot and day, so that each faculty would only teach a particular class at a particular time on particular day.
>
> **Simplicity:** The chosen data structures offer a straightforward and intuitive way to represent and manage faculty assignments and breaks within the timetable generation algorithm.

<u>How We Reached Here?:</u>

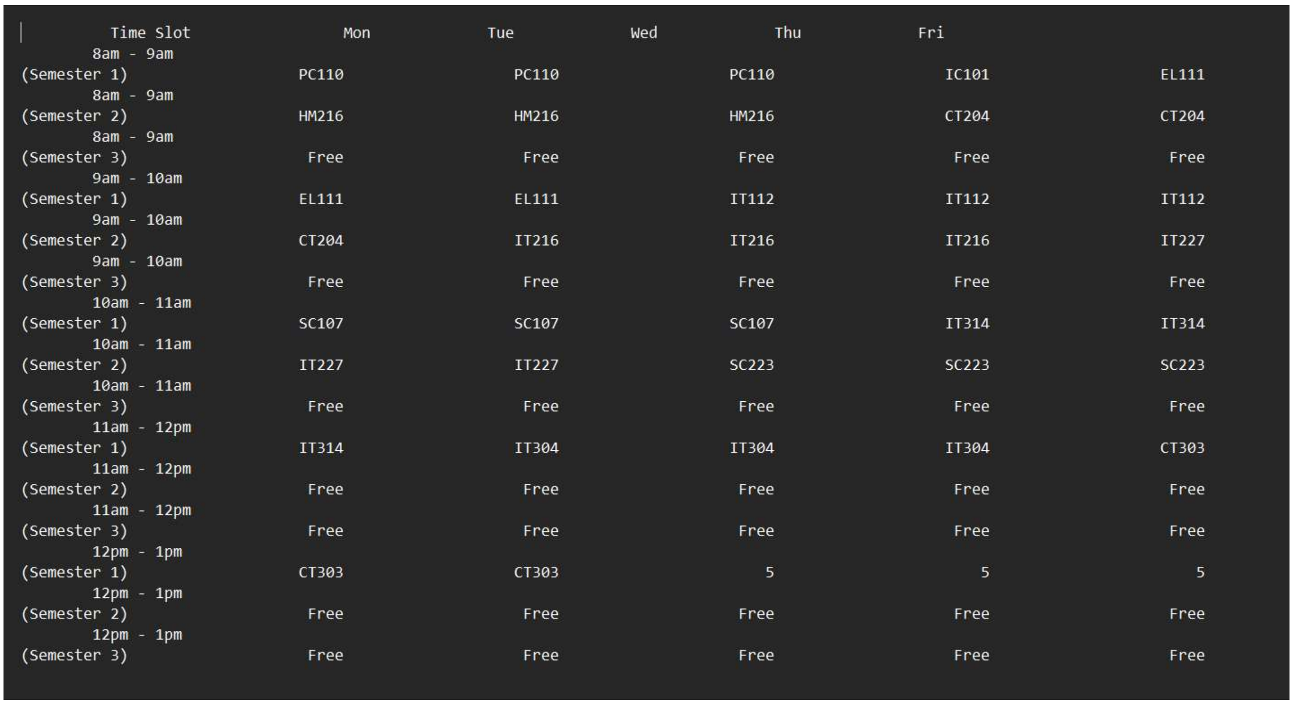
First we started from the basics, and only took input of B Tech ICT branch, that is Section A, and that too for first three semesters. We didn’t even assign the classrooms as it was just the beginning, and we were just trying to figure out the algorithm. We started by adding all the courses sequentially and printing the output on a text file. We got a very congested and bad looking output but things were finally starting to fall in place.

<u>First Output Of Our Program</u>



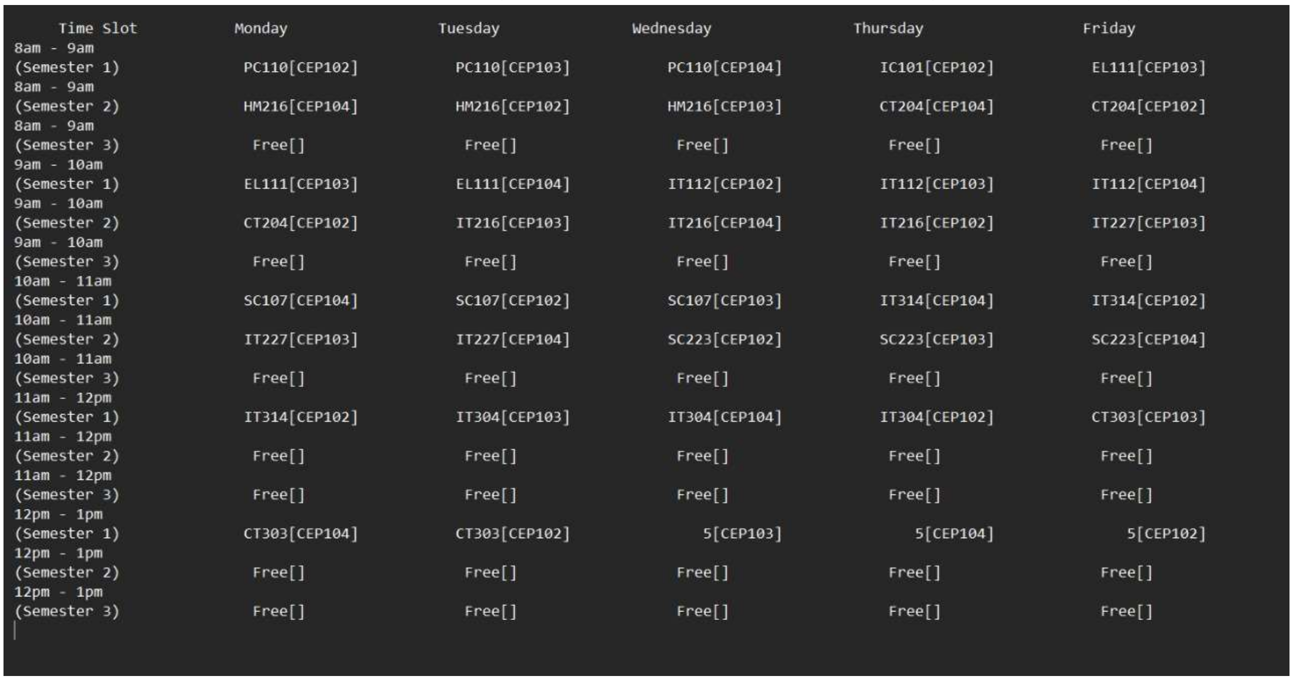
```
        Time Slot              Mon          Tue          Wed          Thu          Fri
  8am - 9am (Semester 1)       Language_and_Literature Language_and_Literature Language_and_Literature  Introduction_to_ICT    Basic_Electronic_Circuits
  8am - 9am (Semester 2)       Science_Technology_and_Society  Science_Technology_and_Society  Science_Technology_and_Society   Signals_and_Systems      Signals_and_Systems
  8am - 9am (Semester 3)                  Free                  Free                  Free                  Free                  Free
 9am - 10am (Semester 1)       Basic_Electronic_Circuits       Basic_Electronic_Circuits       Introduction_to_Programming      Introduction_to_Programming      Introduction_to_Programming

 9am - 10am (Semester 2)        Signals_and_Systems    Design_and_Analysis_of_Algorithms       Design_and_Analysis_of_Algorithms       Design_and_Analysis_of_Algorithms
      Computer_Systems_Programming
 9am - 10am (Semester 3)                  Free                  Free                  Free                  Free                  Free
10am - 11am (Semester 1)               Calculus              Calculus              Calculus    Software_Engineering    Software_Engineering
10am - 11am (Semester 2)       Computer_Systems_Programming    Computer_Systems_Programming         Linear_Algebra         Linear_Algebra         Linear_Algebra
10am - 11am (Semester 3)                  Free                  Free                  Free                  Free                  Free
11am - 12pm (Semester 1)       Software_Engineering       Computer_Networks       Computer_Networks       Computer_Networks    Digital_Communication
11am - 12pm (Semester 2)                  Free                  Free                  Free                  Free                  Free
11am - 12pm (Semester 3)                  Free                  Free                  Free                  Free                  Free
 12pm - 1pm (Semester 1)       Digital_Communication   Digital_Communication                 IE402                 IE402                 IE402
 12pm - 1pm (Semester 2)                  Free                  Free                  Free                  Free                  Free
 12pm - 1pm (Semester 3)                  Free                  Free                  Free                  Free                  Free
```

Later, we tried to improve the formatting of our output (by manually places tabs and spaces and \n wherever we thought the need was) and we got the following result:



```
        Time Slot                Mon              Tue              Wed              Thu              Fri
        8am - 9am
(Semester 1)            PC110               PC110               PC110               IC101               EL111
        8am - 9am
(Semester 2)            HM216               HM216               HM216               CT204               CT204
        8am - 9am
(Semester 3)             Free                Free                Free                Free                Free
        9am - 10am
(Semester 1)            EL111               EL111               IT112               IT112               IT112
        9am - 10am
(Semester 2)            CT204               IT216               IT216               IT216               IT227
        9am - 10am
(Semester 3)             Free                Free                Free                Free                Free
        10am - 11am
(Semester 1)            SC107               SC107               SC107               IT314               IT314
        10am - 11am
(Semester 2)            IT227               IT227               SC223               SC223               SC223
        10am - 11am
(Semester 3)             Free                Free                Free                Free                Free
        11am - 12pm
(Semester 1)            IT314               IT304               IT304               IT304               CT303
        11am - 12pm
(Semester 2)             Free                Free                Free                Free                Free
        11am - 12pm
(Semester 3)             Free                Free                Free                Free                Free
        12pm - 1pm
(Semester 1)            CT303               CT303                   5                   5                   5
        12pm - 1pm
(Semester 2)             Free                Free                Free                Free                Free
        12pm - 1pm
(Semester 3)             Free                Free                Free                Free                Free
```

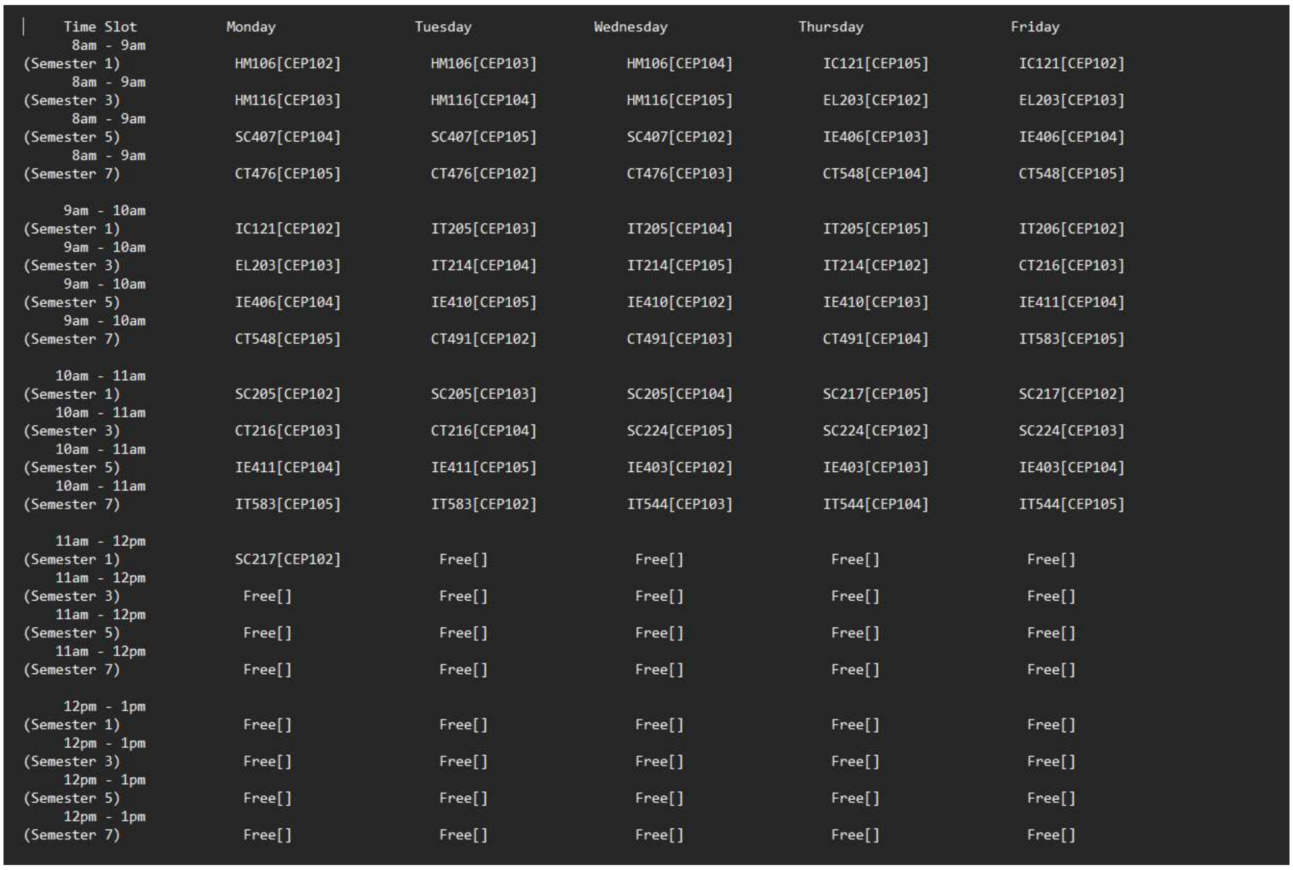
However, it was still difficult in figuring out how to assign the classes, but in the end, we decided that for now, let classes be sequentially shifted row wise. From that we got the following output:



```
    Time Slot          Monday                Tuesday               Wednesday               Thursday               Friday
8am - 9am
(Semester 1)            PC110[CEP102]          PC110[CEP103]          PC110[CEP104]          IC101[CEP102]          EL111[CEP103]
8am - 9am
(Semester 2)            HM216[CEP104]          HM216[CEP102]          HM216[CEP103]          CT204[CEP104]          CT204[CEP102]
8am - 9am
(Semester 3)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
9am - 10am
(Semester 1)            EL111[CEP103]          EL111[CEP104]          IT112[CEP102]          IT112[CEP103]          IT112[CEP104]
9am - 10am
(Semester 2)            CT204[CEP102]          IT216[CEP103]          IT216[CEP104]          IT216[CEP102]          IT227[CEP103]
9am - 10am
(Semester 3)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
10am - 11am
(Semester 1)            SC107[CEP104]          SC107[CEP102]          SC107[CEP103]          IT314[CEP104]          IT314[CEP102]
10am - 11am
(Semester 2)            IT227[CEP103]          IT227[CEP104]          SC223[CEP102]          SC223[CEP103]          SC223[CEP104]
10am - 11am
(Semester 3)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
11am - 12pm
(Semester 1)            IT314[CEP102]          IT304[CEP103]          IT304[CEP104]          IT304[CEP102]          CT303[CEP103]
11am - 12pm
(Semester 2)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
11am - 12pm
(Semester 3)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
12pm - 1pm
(Semester 1)            CT303[CEP104]          CT303[CEP102]              5[CEP103]              5[CEP104]              5[CEP102]
12pm - 1pm
(Semester 2)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
12pm - 1pm
(Semester 3)             Free[]                 Free[]                 Free[]                 Free[]                 Free[]
```

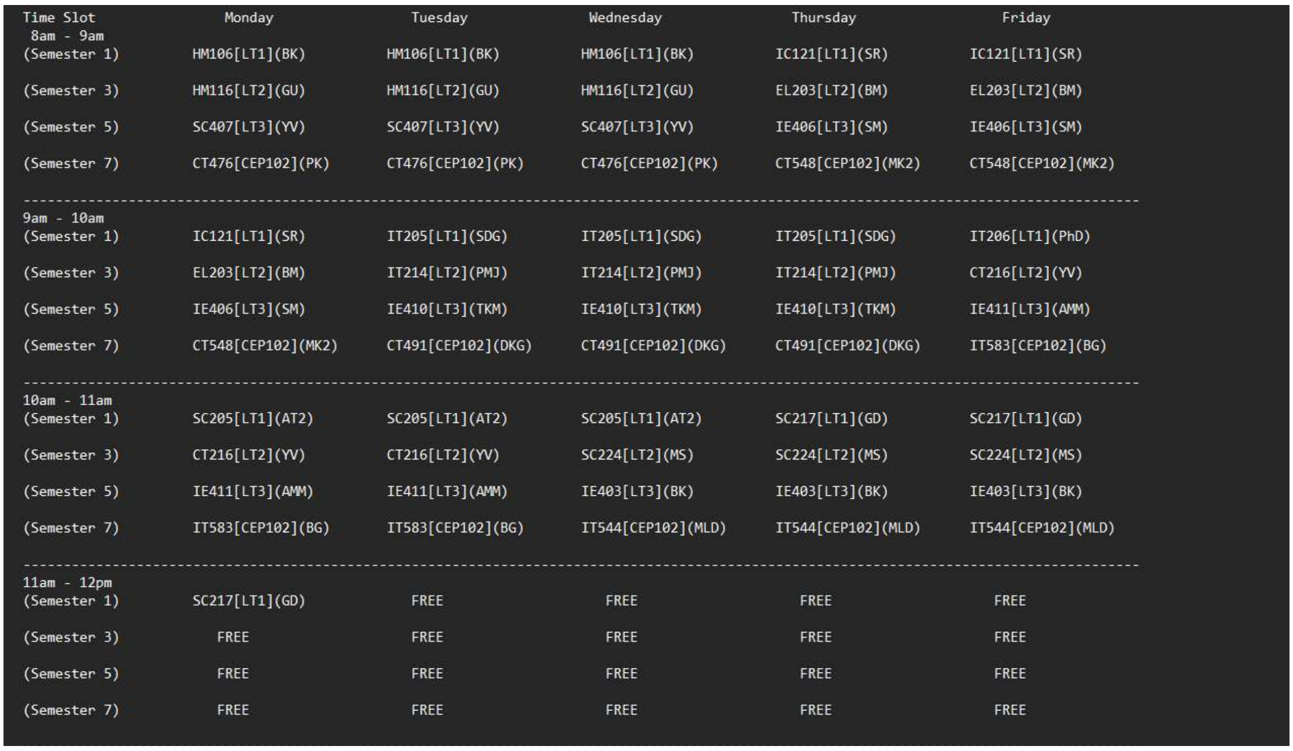
However, another issue caught our eye, which was that semester 3 was never being assigned any classes, which was because our timetable vector did not initially contain a tuple, and was resized in the wrong way. We resized its no of rows to timeslots* 3 instead of 9.This was done to ensure that each class and course of a particular semester was duly assigned to a specific box in our timetable grid. We also added the last semester by choosing some of the electives by ourselves and resized rows of timetable vector to timeslots*4.

This was the new output we got:



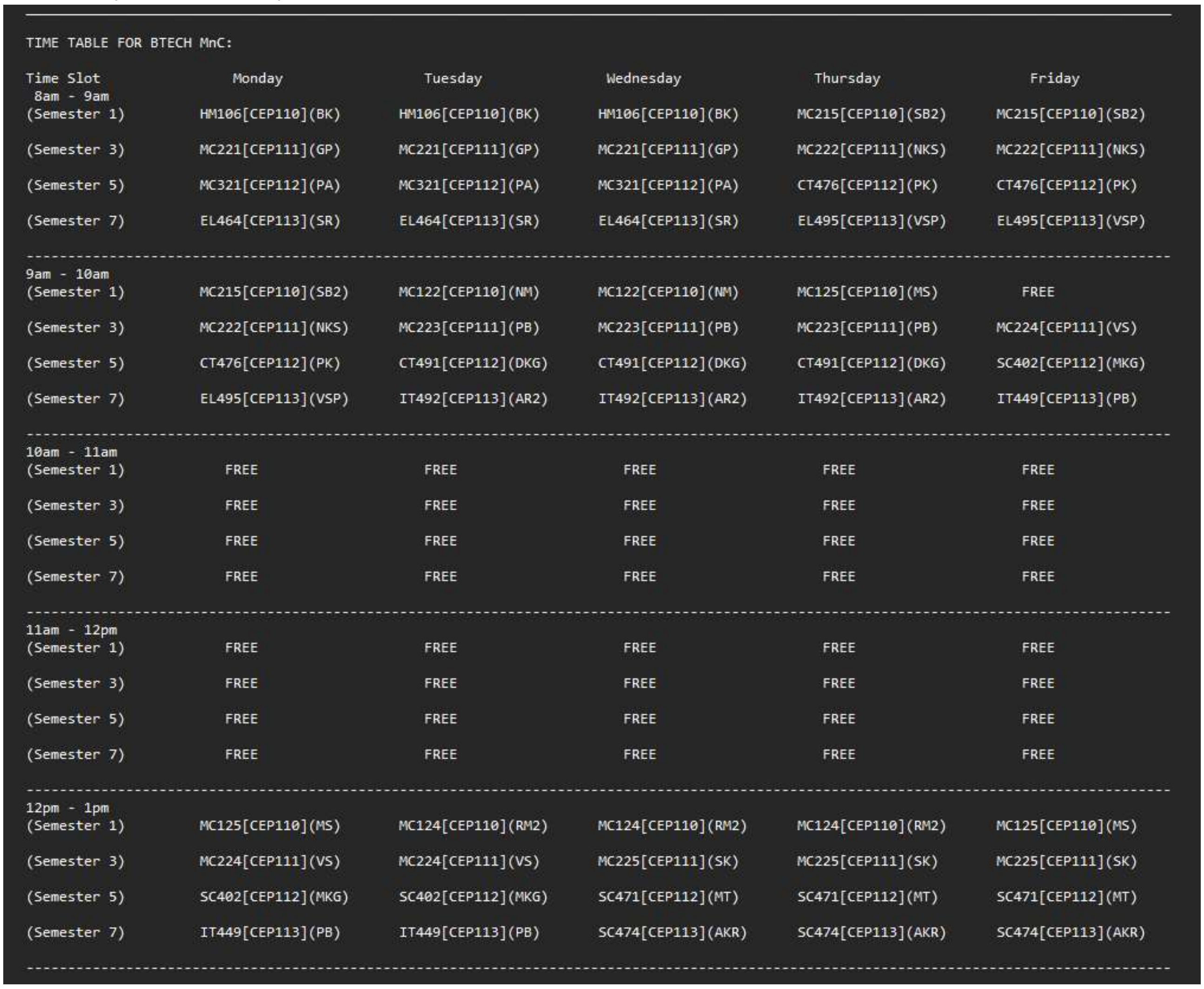
```
|     Time Slot          Monday              Tuesday             Wednesday             Thursday              Friday
       8am - 9am
 (Semester 1)           HM106[CEP102]        HM106[CEP103]        HM106[CEP104]         IC121[CEP105]         IC121[CEP102]
       8am - 9am
 (Semester 3)           HM116[CEP103]        HM116[CEP104]        HM116[CEP105]         EL203[CEP102]         EL203[CEP103]
       8am - 9am
 (Semester 5)           SC407[CEP104]        SC407[CEP105]        SC407[CEP102]         IE406[CEP103]         IE406[CEP104]
       8am - 9am
 (Semester 7)           CT476[CEP105]        CT476[CEP102]        CT476[CEP103]         CT548[CEP104]         CT548[CEP105]

      9am - 10am
 (Semester 1)           IC121[CEP102]        IT205[CEP103]        IT205[CEP104]         IT205[CEP105]         IT206[CEP102]
      9am - 10am
 (Semester 3)           EL203[CEP103]        IT214[CEP104]        IT214[CEP105]         IT214[CEP102]         CT216[CEP103]
      9am - 10am
 (Semester 5)           IE406[CEP104]        IE410[CEP105]        IE410[CEP102]         IE410[CEP103]         IE411[CEP104]
      9am - 10am
 (Semester 7)           CT548[CEP105]        CT491[CEP102]        CT491[CEP103]         CT491[CEP104]         IT583[CEP105]

     10am - 11am
 (Semester 1)           SC205[CEP102]        SC205[CEP103]        SC205[CEP104]         SC217[CEP105]         SC217[CEP102]
     10am - 11am
 (Semester 3)           CT216[CEP103]        CT216[CEP104]        SC224[CEP105]         SC224[CEP102]         SC224[CEP103]
     10am - 11am
 (Semester 5)           IE411[CEP104]        IE411[CEP105]        IE403[CEP102]         IE403[CEP103]         IE403[CEP104]
     10am - 11am
 (Semester 7)           IT583[CEP105]        IT583[CEP102]        IT544[CEP103]         IT544[CEP104]         IT544[CEP105]

     11am - 12pm
 (Semester 1)           SC217[CEP102]         Free[]               Free[]                Free[]                Free[]
     11am - 12pm
 (Semester 3)            Free[]               Free[]               Free[]                Free[]                Free[]
     11am - 12pm
 (Semester 5)            Free[]               Free[]               Free[]                Free[]                Free[]
     11am - 12pm
 (Semester 7)            Free[]               Free[]               Free[]                Free[]                Free[]

      12pm - 1pm
 (Semester 1)            Free[]               Free[]               Free[]                Free[]                Free[]
      12pm - 1pm
 (Semester 3)            Free[]               Free[]               Free[]                Free[]                Free[]
      12pm - 1pm
 (Semester 5)            Free[]               Free[]               Free[]                Free[]                Free[]
      12pm - 1pm
 (Semester 7)            Free[]               Free[]               Free[]                Free[]                Free[]
```

Next task was to ensure that classes of a particular course are always held in same class as is the norm. Now there was a trade-off, if we wanted that to happen, classes of a particular semester will always be held in the same class which was done after initial reluctance:



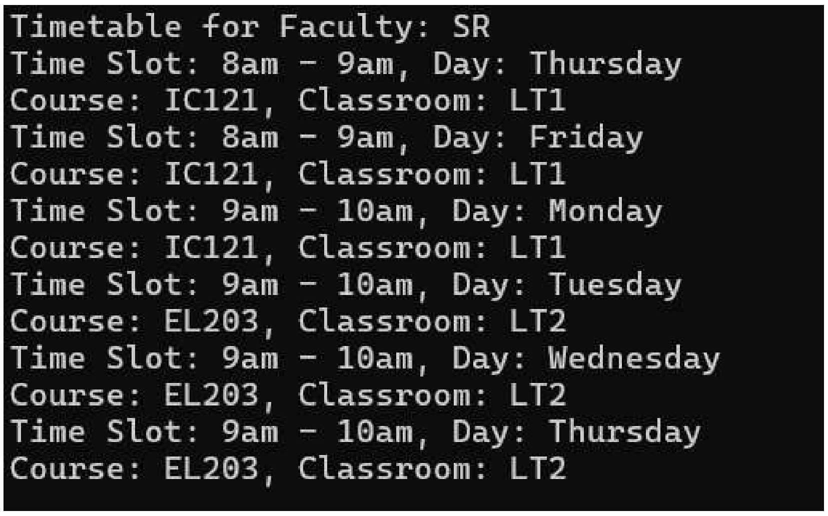
```
Time Slot              Monday                Tuesday              Wednesday              Thursday              Friday
 8am - 9am
(Semester 1)         HM106[LT1](BK)        HM106[LT1](BK)        HM106[LT1](BK)        IC121[LT1](SR)        IC121[LT1](SR)

(Semester 3)         HM116[LT2](GU)        HM116[LT2](GU)        HM116[LT2](GU)        EL203[LT2](BM)        EL203[LT2](BM)

(Semester 5)         SC407[LT3](YV)        SC407[LT3](YV)        SC407[LT3](YV)        IE406[LT3](SM)        IE406[LT3](SM)

(Semester 7)         CT476[CEP102](PK)     CT476[CEP102](PK)     CT476[CEP102](PK)     CT548[CEP102](MK2)    CT548[CEP102](MK2)

------------------------------------------------------------------------------------------------------------------------
9am - 10am
(Semester 1)         IC121[LT1](SR)        IT205[LT1](SDG)       IT205[LT1](SDG)       IT205[LT1](SDG)       IT206[LT1](PhD)

(Semester 3)         EL203[LT2](BM)        IT214[LT2](PMJ)       IT214[LT2](PMJ)       IT214[LT2](PMJ)       CT216[LT2](YV)

(Semester 5)         IE406[LT3](SM)        IE410[LT3](TKM)       IE410[LT3](TKM)       IE410[LT3](TKM)       IE411[LT3](AMM)

(Semester 7)         CT548[CEP102](MK2)    CT491[CEP102](DKG)    CT491[CEP102](DKG)    CT491[CEP102](DKG)    IT583[CEP102](BG)

------------------------------------------------------------------------------------------------------------------------
10am - 11am
(Semester 1)         SC205[LT1](AT2)       SC205[LT1](AT2)       SC205[LT1](AT2)       SC217[LT1](GD)        SC217[LT1](GD)

(Semester 3)         CT216[LT2](YV)        CT216[LT2](YV)        SC224[LT2](MS)        SC224[LT2](MS)        SC224[LT2](MS)

(Semester 5)         IE411[LT3](AMM)       IE411[LT3](AMM)       IE403[LT3](BK)        IE403[LT3](BK)        IE403[LT3](BK)

(Semester 7)         IT583[CEP102](BG)     IT583[CEP102](BG)     IT544[CEP102](MLD)    IT544[CEP102](MLD)    IT544[CEP102](MLD)

------------------------------------------------------------------------------------------------------------------------
11am - 12pm
(Semester 1)         SC217[LT1](GD)            FREE                  FREE                  FREE                  FREE

(Semester 3)            FREE                   FREE                  FREE                  FREE                  FREE

(Semester 5)            FREE                   FREE                  FREE                  FREE                  FREE

(Semester 7)            FREE                   FREE                  FREE                  FREE                  FREE
```

Next, we decided to include MnC branch in our output. The first time was done by creating two objects of our class TTGen and seperately creating timetable of each branch as shown(ICT previously, MnC below):



```
TIME TABLE FOR BTECH MnC:

Time Slot              Monday              Tuesday             Wednesday           Thursday            Friday
 8am - 9am
(Semester 1)        HM106[CEP110](BK)   HM106[CEP110](BK)   HM106[CEP110](BK)   MC215[CEP110](SB2)  MC215[CEP110](SB2)

(Semester 3)        MC221[CEP111](GP)   MC221[CEP111](GP)   MC221[CEP111](GP)   MC222[CEP111](NKS)  MC222[CEP111](NKS)

(Semester 5)        MC321[CEP112](PA)   MC321[CEP112](PA)   MC321[CEP112](PA)   CT476[CEP112](PK)   CT476[CEP112](PK)

(Semester 7)        EL464[CEP113](SR)   EL464[CEP113](SR)   EL464[CEP113](SR)   EL495[CEP113](VSP)  EL495[CEP113](VSP)

------------------------------------------------------------------------------------------------------------------------
9am - 10am
(Semester 1)        MC215[CEP110](SB2)  MC122[CEP110](NM)   MC122[CEP110](NM)   MC125[CEP110](MS)      FREE

(Semester 3)        MC222[CEP111](NKS)  MC223[CEP111](PB)   MC223[CEP111](PB)   MC223[CEP111](PB)   MC224[CEP111](VS)

(Semester 5)        CT476[CEP112](PK)   CT491[CEP112](DKG)  CT491[CEP112](DKG)  CT491[CEP112](DKG)  SC402[CEP112](MKG)

(Semester 7)        EL495[CEP113](VSP)  IT492[CEP113](AR2)  IT492[CEP113](AR2)  IT492[CEP113](AR2)  IT449[CEP113](PB)

------------------------------------------------------------------------------------------------------------------------
10am - 11am
(Semester 1)           FREE                FREE                FREE                FREE                FREE

(Semester 3)           FREE                FREE                FREE                FREE                FREE

(Semester 5)           FREE                FREE                FREE                FREE                FREE

(Semester 7)           FREE                FREE                FREE                FREE                FREE

------------------------------------------------------------------------------------------------------------------------
11am - 12pm
(Semester 1)           FREE                FREE                FREE                FREE                FREE

(Semester 3)           FREE                FREE                FREE                FREE                FREE

(Semester 5)           FREE                FREE                FREE                FREE                FREE

(Semester 7)           FREE                FREE                FREE                FREE                FREE

------------------------------------------------------------------------------------------------------------------------
12pm - 1pm
(Semester 1)        MC125[CEP110](MS)   MC124[CEP110](RM2)  MC124[CEP110](RM2)  MC124[CEP110](RM2)  MC125[CEP110](MS)

(Semester 3)        MC224[CEP111](VS)   MC224[CEP111](VS)   MC225[CEP111](SK)   MC225[CEP111](SK)   MC225[CEP111](SK)

(Semester 5)        SC402[CEP112](MKG)  SC402[CEP112](MKG)  SC471[CEP112](MT)   SC471[CEP112](MT)   SC471[CEP112](MT)

(Semester 7)        IT449[CEP113](PB)   IT449[CEP113](PB)   SC474[CEP113](AKR)  SC474[CEP113](AKR)  SC474[CEP113](AKR)

------------------------------------------------------------------------------------------------------------------------
```

However, a new problem arised, which was that we could not check for faculty clashes between ICT and MnC as both were generated through different objects and faculty check was done individually done for both and not for each other together. We also added a faculty timetable output initially in cmd which would output timetable for a specific faculty:



```
Timetable for Faculty: SR
Time Slot: 8am - 9am, Day: Thursday
Course: IC121, Classroom: LT1
Time Slot: 8am - 9am, Day: Friday
Course: IC121, Classroom: LT1
Time Slot: 9am - 10am, Day: Monday
Course: IC121, Classroom: LT1
Time Slot: 9am - 10am, Day: Tuesday
Course: EL203, Classroom: LT2
Time Slot: 9am - 10am, Day: Wednesday
Course: EL203, Classroom: LT2
Time Slot: 9am - 10am, Day: Thursday
Course: EL203, Classroom: LT2
```

Next, we decided to include ICT+CS(sec B) in our timetable, and did it by writing secA and secB in the courses respectively. Electives were choses randomly by us for the time being.
The link to that output is:
https://github.com/Shamit2105/Timetable/commit/e8b7d70e71983b915e2acea74dd41e6fbecb3ad8

After that, we added MnC branch as every third row and timetable for entire B Tech was completed. M Tech ICT was added and the output was:
https://github.com/Shamit2105/Timetable/commit/c79d1fa86d427f286485a4193abe67772b4284fc

Further branches were added and finally we got the desired output, all electives for semesters 4 and 5 were displayed but for semesters 6,7 and 8 we randomly picked out some electives as we were confused about how to display all of them.

https://github.com/Shamit2105/Timetable/blob/main/second_version/timetable.txt
This is the link to our final timetable.txt file for the Autumn Semester(Odd semester) and as you can see, for eg: SDG (Sourish Sir) has ! mark after his name, so he will not be getting assigned continuous courses. Same with SB2. Also courses having (ELE) are electives and for semester 5 and semester 7, there are 2 rows for ICT + CS, as they have lot of common electives and if a course is specifically for CS students, it has (CS) marked in brackets.

Here is output for winter semester where SDG sir, SR mam, and PD mam will always have a break between 2 classes on any given day.
https://github.com/Shamit2105/Timetable/blob/main/second_version/timetable2.txt

The github link of our repository has 2 versions of our program. The first version is the basic version of our program as mentioned earlier in the document. The folder second_version has the final version of our program with outputs and inputs of both winter and autumn semesters, along with faculty output and classroom output in .txt format.

Explanation Of Main Algorithm:

We start by initializing various data structures required for managing the timetable generation process. These include lectures_assigned, faculty_assigned, and last_faculty, which help us keep track of course assignments, faculty availability, and the last faculty member who taught in each time slot and day, respectively.

Moving on, we traverse through each time slot, representing different hours of the day from 8 am to 1 pm. Within this outer loop, we proceed to iterate over each semester, representing different sets of courses. This nested iteration helps us cover all possible combinations of time slots and semesters.

As we loop through each semester, we further iterate over each day of the week, starting from Monday and ending at Friday. This allows us to assign courses to specific days, ensuring a balanced distribution throughout the week.

During each iteration, we check whether any course needs to be assigned for the current time slot, semester, and day. We verify faculty availability, looking for clashes with previously assigned courses and ensuring breaks for faculty members who requested them.

Upon finding a suitable course to assign, we update the timetable accordingly, marking the course, classroom, and faculty for the given time slot, semester, and day. Additionally, we update our data structures to reflect the changes in course assignments and faculty availability.

In cases where no course can be assigned for a particular time slot, we designate it as "Free" in the timetable to indicate its availability.

We add the slots in the index [22*i + semester][j] where j stands for a particular day, while the first expression is a unique index within the timetable vector for each combination of time slot (i), semester (semester), and day, ensuring that each course assignment is stored at the correct position in the timetable grid.

Now, why multiply by 22 and add semester?

This calculation ensures that each combination of time slot and semester corresponds to a unique index in the timetable vector. Since there are 5 time slots and 22 semesters, we need a way to calculate a unique index for each combination. Multiplying the number of time slots by the total number of semesters (22 * i) provides a base index for the time slot, and then adding semester allows us to differentiate between different semesters within that time slot.

This expression effectively maps each time slot, semester, and day combination to a specific location in the timetable data structure.

Once all iterations are complete, we write the generated timetable to a text file named "timetable.txt", ensuring proper formatting and organization for readability.

Constraints Checked:

i) Checked that there are no faculty clashes at a given time on a given day.(Lines 146-150 in GITHUB CODE)
ii) Provided breaks to faculty if they need it(having ! as last character of faculty's name means no consecutive classes for faculty).(Lines 152 to 160 in GITHUB code)
iii) Electives for semester 4,5 and 6.(Adjusted in input)
iv) Outputted timetable for faculty as well as classroom.
v) If multiple faculties teach same courses, adjusted it accordingly in our timetable(by assigning sections in input itself).
vi) If same course taught by multiple faculties, then adjusted it accordingly(by assigning sections in input itself).
vii) Ensured every semester has at-least one free period every day.

Improvements:

Even though we are getting a considerably good output from our program, these are the changes that we will further try to make just for the betterment of the program (even after submission) just for our learning of the course:
1) Currently, each semester is sitting in fix classrooms for every lecture, which we will try to change.
2) We will try to adjust electives of semester 7 and 8 in our input and join them whenever they are linked with M Tech Courses.
3) Try to reduce the time complexity and space complexity of our codes.