**Timetable Scheduling System**

Author:

MUHAMMAD SAAD KHAN

**Overview**

The provided C++ program simulates a timetable management system for a university. It allows for the loading of data from a CSV file, adjusting the timetable, handling teacher conflicts, and updating the timetable with various functionalities such as manual updates, automatic adjustments, searching for teachers, and saving the updated timetable to a file.

**Program Features**

1. **Load Timetable Data**: The program loads data from a CSV file containing teacher names, subjects, room numbers, lab information, and credit hours.
2. **Automatic Timetable Adjustment**: The program adjusts the timetable to avoid consecutive lab sessions and ensure that subjects are assigned to sections with no teacher conflicts.
3. **Manual Timetable Update**: The user can manually edit the timetable.
4. **Search for Teacher's Classes**: The system allows users to search for all classes taught by a specific teacher.
5. **Backup Timetable**: Users can backup the adjusted timetable to an Excel-like CSV file.
6. **Teacher Conflict Management**: The system tracks and handles teacher conflicts, ensuring no teacher is assigned to multiple classes at the same time.

**Data Structures**

**Node Structure**

struct Node {

string T\_Name; // Teacher's Name

string Subject; // Subject Name

int Room\_no; // Room Number

bool HasLab; // Lab Session (True/False)

int CreditHour; // Credit Hours

int OriginalCreditHour; // Original Credit Hours

Node\* next; // Pointer to next node in circular doubly linked list

Node\* prev; // Pointer to previous node in circular doubly linked list

};

**Purpose**: The node structure represents a single subject-teacher pair and is used to create a circular doubly linked list to store the timetable entries.

* **Fields**:
  + T\_Name: Teacher's name.
  + Subject: Name of the subject.
  + Room\_no: The room number for the class.
  + HasLab: Boolean indicating if the subject is a lab session.
  + CreditHour: The number of credit hours for the subject.
  + OriginalCreditHour: Keeps track of the original credit hours to reset after adjustments.
  + next: Pointer to the next node in the circular list.
  + prev: Pointer to the previous node in the circular list.

**TeacherConflict Structure**

struct TeacherConflict {

string T\_Name; // Teacher's Name

int day; // Day of the conflict

int slot; // Time slot of the conflict

};

* **Purpose**: This structure stores the conflicts for a teacher (if they are scheduled for more than one class at the same time).

**TeacherSchedule Structure**

struct TeacherSchedule {

string T\_Name; // Teacher's Name

TeacherConflict conflicts[Day \* Slot]; // Array to store conflicts

int count = 0; // Number of conflicts for this teacher

};

* **Purpose**: This structure holds the teacher's name and the conflicts (if any) that arise in the timetable for that teacher.
* **Fields**:
  + T\_Name: Teacher's name.
  + conflicts: Array of conflicts for each teacher.
  + count: Number of conflicts for this teacher.

**Functions**

**insertTeacherConflict**

void insertTeacherConflict(const string& t\_name, int day, int slot);

* **Purpose**: Adds a conflict for the teacher in the teacherSchedules array.
* **Parameters**:
  + t\_name: Teacher's name.
  + day: Day of the conflict.
  + slot: Slot of the conflict.

**hasTeacherConflict**

cpp

Copy code

bool hasTeacherConflict(const string& t\_name, int day, int slot);

* **Purpose**: Checks if a teacher has a conflict on a particular day and slot.
* **Parameters**:
  + t\_name: Teacher's name.
  + day: Day to check.
  + slot: Slot to check.
* **Returns**: true if the teacher has a conflict, false otherwise.

**insertNode**

void insertNode(string t\_name, string subject, int room\_no, bool hasLab, int creditHour);

* **Purpose**: Inserts a new node in the circular doubly linked list.
* **Parameters**:
  + t\_name: Teacher's name.
  + subject: Subject name.
  + room\_no: Room number for the class.
  + hasLab: Indicates whether the subject is a lab.
  + creditHour: Number of credit hours for the subject.

**loadDataFromCSV**

void loadDataFromCSV(const string& fileName);

* **Purpose**: Loads timetable data from a CSV file.
* **Parameters**:
  + fileName: Name of the CSV file to load the data from.

**shuffleListAvoidingConsecutiveLabs**

void shuffleListAvoidingConsecutiveLabs();

* **Purpose**: Shuffles the timetable while ensuring no consecutive lab sessions are assigned.

**auto\_adjust**

void auto\_adjust();

* **Purpose**: Automatically adjusts the timetable to avoid teacher conflicts and assigns subjects to slots, avoiding consecutive lab sessions.

**printTimeTable**

void printTimeTable();

* **Purpose**: Prints the current state of the adjusted timetable.

**saveAdjustedTimetableToCSV**

void saveAdjustedTimetableToCSV(const string& fileName);

* **Purpose**: Saves the adjusted timetable to a CSV file.
* **Parameters**:
  + fileName: The name of the file where the timetable will be saved.

**saveTimetableToExcel**

void saveTimetableToExcel(const string& filename);

* **Purpose**: Saves the current timetable to an Excel-like CSV file.
* **Parameters**:
  + filename: The name of the file where the timetable will be backed up.

**toLowerCase**

string toLowerCase(const string& str);

* **Purpose**: Converts a string to lowercase.
* **Parameters**:
  + str: The string to convert.
* **Returns**: The converted lowercase string.

**searchTeacherClassesInFile**

void searchTeacherClassesInFile(const string& filename);

* **Purpose**: Searches for a teacher's classes in a CSV file and prints out the details.
* **Parameters**:
  + filename: The name of the CSV file containing the timetable data.

**editTimetable**

void editTimetable();

* **Purpose**: Allows the user to edit the timetable by updating a specific section, day, and slot with new subject details.

**Main Menu Options**

* **Option 1**: Backup Timetable - Saves the current timetable to a backup CSV file.
* **Option 2**: Search for Specific Teacher - Allows searching for classes taught by a specific teacher.
* **Option 3**: Edit/Update Timetable - Lets the user edit a specific entry in the timetable.
* **Option 4**: Exit - Exits the program.

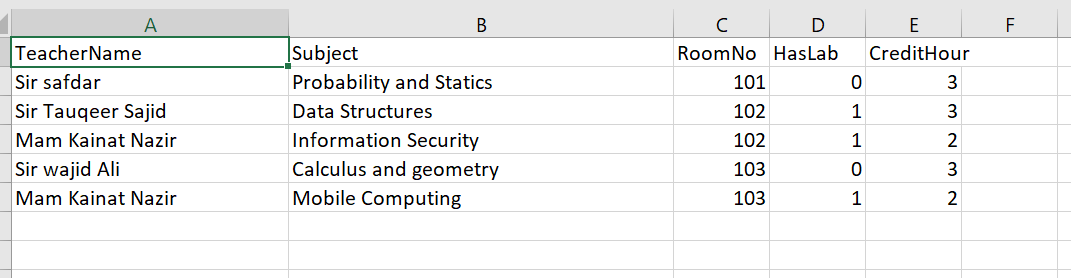
**File Formats**

**CSV File Format (Input)**

The input CSV file should have the following format:

Teacher Name,Subject,Room Number,Lab (1 for lab, 0 for non-lab),Credit Hour

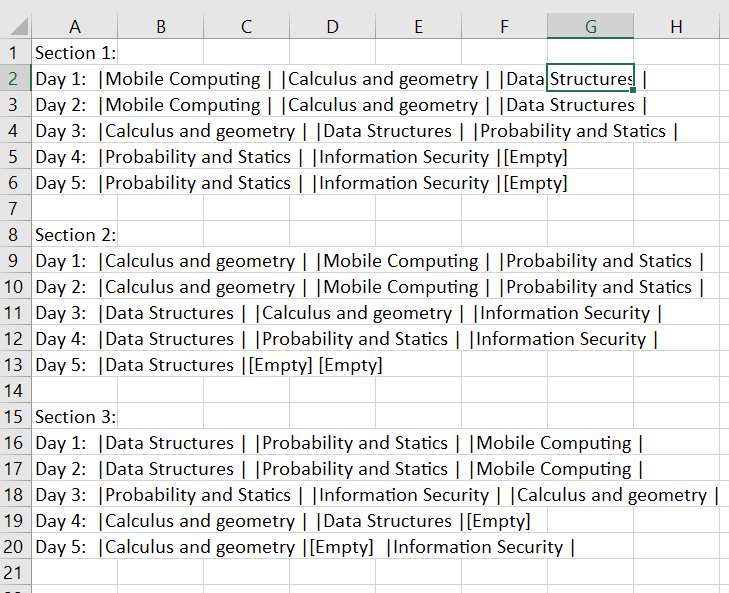
Example:

**CSV File Format (Output)**

The output CSV for the adjusted timetable is in the format:

Section,Day,Slot,Subject

Example:

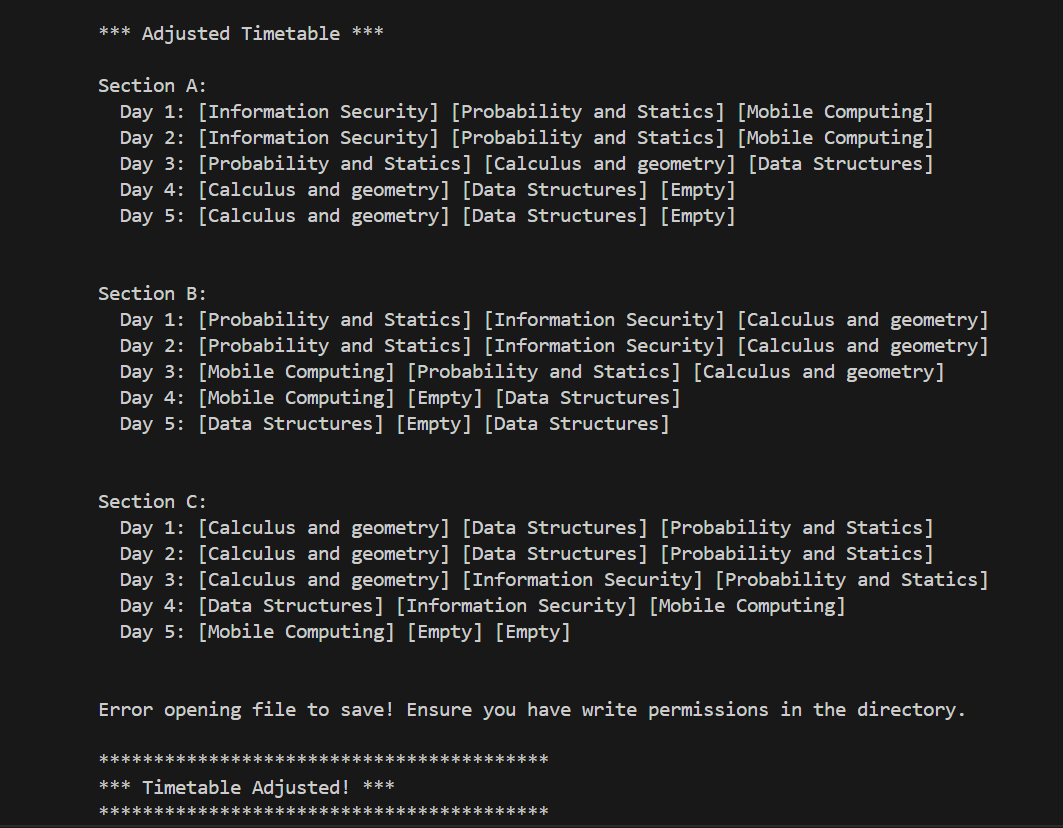


**Conclusion**

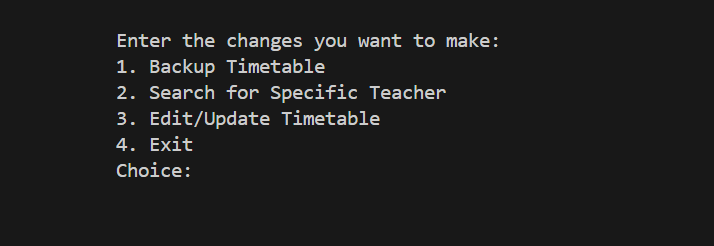
This program provides a comprehensive solution for managing and adjusting university timetables. It ensures that the timetable is optimized by avoiding conflicts and consecutive lab sessions while allowing for manual updates and data persistence in CSV format.

SCREENSHOTS:

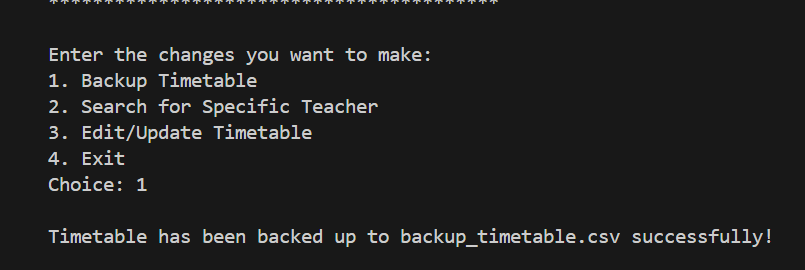
## Adjusted Timetable:

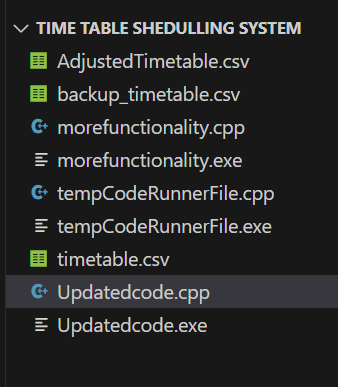


## MENU:

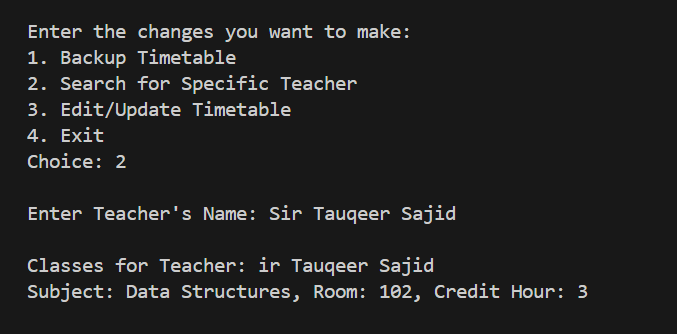


## BACKUP TIMETABLE:





## SEARCHING TIMETABLE:



## EDIT/UPDATE TIMETABLE:

