**C++ Assignment-3**

**Program overview-**

This program illustrates a congregation manager where in many congregations can be held simulatenously at different venues with each venue having many events.

**1. Time Class**

**Purpose:**  
Represents a specific time with hours and minutes.

**Attributes:**

* int hour: The hour component of the time.
* int minute: The minute component of the time.

**Constructors:**

* Time(int hour, int minute): Initializes the time object with specific hour and minute.
* Time(): Default constructor that initializes the time to 00:00.

**Methods:**

* **Comparison Operators:>= ,> and ==**
* **Arithmetic Operator: -**
* **Output Stream Operator:<<**

**2. Date Class**

**Purpose:**  
Represents a specific date with year, month, and day.

**Attributes:**

* int year: The year component of the date.
* int month: The month component of the date.
* int day: The day component of the date.

**Constructors:**

* Date(int year, int month, int day): Initializes the date object with specific year, month, and day.
* Date(): Default constructor that initializes the date to 0000-00-00.

**Methods:**

* **Comparison Operators: >=,<=,>,<,==**
* **Increment Operator: Date++**
* **Arithmetic Operator:** int operator-(const Date &secondobj) const
* **Output Stream Operator:<<**

**3. Event Class**

**Purpose:**  
Represents an event scheduled on a specific date with a start time, end time, and a name.

**Attributes:**

* string name: The name of the event.
* Date date: The date of the event.
* Time start\_hour: The start time of the event.
* Time end\_hour: The end time of the event.

**Constructors:**

* Event(Date date, Time start\_hour, Time end\_hour, string eventName): Initializes the event object with specific date, start time, end time, and name.

**4. Day Class**

**Purpose:**  
Represents a day that can hold multiple events.

**Attributes:**

* Date date: The date associated with the day.
* int count\_events: The number of events scheduled on this day.
* vector<Event \*> day: A vector of pointers to events scheduled on this day.

**Constructors:**

* Day(Date date): Initializes the day with a specific date and an empty list of events.

**Methods:**

* int addEvent(Time start\_hour, Time end\_hour, string eventName): Adds an event to the day if there are no conflicts with other events. Returns 1 if successful, -1 if there is a conflict.
* int deleteEvent(Time start\_hour, string eventName): Deletes an event from the day based on the event's start time and name. Returns 1 if successful, -1 if the event was not found.
* int showEvents(): Displays all events scheduled on this day, showing the event name, start time, and end time.

**5. Venue Class**

**Purpose:**  
Represents a venue that can host multiple events and conventions.

**Attributes:**

* static vector<Venue \*> venues: A static vector holding pointers to all created venue objects.
* string name: The name of the venue.
* string country: The country where the venue is located.
* vector<string> location: A vector representing the hierarchical location of the venue (e.g., city, state, district).
* int capacity: The capacity of the venue.
* vector<Day \*> calendar: A vector of pointers to Day objects representing the schedule of events at the venue.
* vector<Convention \*> conventions\_: A vector of pointers to conventions that have reserved the venue.

**Constructors:**

* Venue(string venueName, string country, vector<string> location, int capacity): Initializes a venue with specific name, country, location, and capacity.
* Venue(): Default constructor that initializes an empty venue object.

**Methods:**

* int isValidReservation(Date startDate, Date endDate): Checks if the venue is available for reservation between the given start and end dates. Returns 1 if valid, -1 if not.
* static int showVenues(vector<string> &location\_input): Displays all venues matching the given location input. The input can be partial, and it will match the hierarchical location fields accordingly.
* static int venueSearch(string venu, string country, Venue \*&p): Searches for a venue by name and country, and if found, assigns the pointer to p.
* int showCalendar(Date startDate, Date endDate): Displays the schedule of events at the venue between the given start and end dates.
* int showEvents(Date date): Displays all events scheduled on a specific date at the venue.

**6. Convention Class**

**Purpose:**  
Represents a convention that can reserve multiple venues and schedule multiple events.

**Attributes:**

* static vector<Convention \*> conventions: A static vector holding pointers to all created convention objects.
* vector<Venue \*> venues: A vector of pointers to venues reserved by the convention.
* string type: The type of the convention (e.g., conference, concert).
* string name: The name of the convention.
* Date startDate: The start date of the convention.
* Date endDate: The end date of the convention.

**Constructors:**

* Convention(string name, string type, Date startDate, Date endDate): Initializes a convention with a specific name, type, start date, and end date.
* Convention(): Default constructor that initializes an empty convention object.

**Methods:**

* static int ConventionSearch(string name, Convention \*&p): Searches for a convention by name, and if found, assigns the pointer to p.
* int reserveVenue(Venue &venu): Reserves a venue for the convention if the venue is available. Returns 1 if successful, -1 if the venue is unavailable.
* int freeVenue(Venue &venu): Releases a reserved venue, removing all events scheduled by the convention at that venue. Returns 1 if successful, -1 if the venue was not reserved by the convention.
* int showReserved(): Displays all venues reserved by the convention.
* int addEvent(Venue &venu, Date date, Time start\_hour, Time end\_hour, string eventName): Adds an event to a reserved venue if there are no conflicts with other events. Returns 1 if successful, -1 if there is a conflict or if the venue is not reserved by the convention.
* int deleteEvent(Venue &venu, Date date, Time start\_hour, string eventName): Deletes an event from a reserved venue. Returns 1 if successful, -1 if the event or venue was not found.
* int showCalendar(Venue &venu): Displays the schedule of events for a reserved venue between the start and end dates of the convention.

**7. Global Functions**

**Purpose:**  
These functions provide utility operations that are used throughout the program.

* **int getdetails(vector<string> &a)**: Extracts location details from user input, considering special cases like quoted strings and numbers.
* **int daysInYear(int year, int month, int day)**: Calculates the day of the year for a given date.
* **bool isLeapYear(int year)**: Checks if a given year is a leap year.
* **int daysInMonth(int year, int month)**: Returns the number of days in a given month for a given year, accounting for leap years.
* **int isValidAddress(string address, vector<string> &, bool flag = true)**: Validates an address string, splitting it into components.

Class Relationship Diagram:-

