EGC 211 Programming II
C++ Lab 3
19th August 2024

This assignment will build upon the work you did in Assignment 2 to add some new features to your C++ program. This exercise should be done individually – no groups or sharing of code is allowed.

Submission deadline: Aug 28, 11:59 pm

Background: Your company wants to generalize the program you built for Paris Olympics 2024 so that it can handle several large-scale congregations such as Games, Concerts, Conferences and Conventions in future. This requires:

1. The ability to handle multiple congregations simultaneously
2. The ability to add / delete venues and events to each congregation individually
3. The ability to handle a proper venue address of the form "location-address:city:state:postal-code:country". Note that a single congregation such as a Cricket World Cup may hold matches at venues in multiple countries.
4. The ability to handle proper date format such as "YYYY-MM-DD" for an event date and time format as "hh:mm" for event start and end-time. You may assume that 0 <= hh <= 23, 0 <= mm <= 59 and start / end times must align with a 15 min interval (00, 15, 30 or 45 mins after any hour between 00 and 23. For example, start / end time of 13:45 or 00:00 is valid but 11:04 or 22:35 is NOT because these do not align with a 15-min boundary.
5. Constraints on event-scheduling:
    a. An event must be held for a minimum of 30 min duration from start to end.
    b. A venue needs at least 30 mins of cleaning time between 2 consecutive events – even if one event ends on one day and the next one starts on the following day. For example, if any event ends at 23:45 hours on 2024-09-16 at a certain venue, the next event at the same venue cannot start before 00:15 hours on 2024-09-17.

Exercise:

1. Applying object-oriented design methodology, document and add any additional entities and their inter-relationships to your UML diagrams. Briefly explain your design thinking in a design document (3 pages max including diagrams).
2. Implement new classes needed and program changes required for the new features in C++. Please see below for new commands and changes to the commands used earlier.

    A. You need to support a new set of commands to add / delete congregations to your system.
        i. addCongregation <name-string> <type-string> <start-date-string> <end-date-string>
        [prints 0 for success, -1 for a failure. Type of congregation may be one of "Concert", "Games", "Convention", "Conference" for now. Dates should be entered as "YYYY-MM-DD". Assume that congregation names are unique across the system, e.g. "Summer Olympics 2024" or "US Open Tennis

2024", "Wimbledon 2025". Dates should be validated, e.g. "2025-02-29" or "2024-09-31" are invalid dates.]

ii. deleteCongregation <name-string>
[prints 0 for success, -1 for a failure. This command also deletes associated reservations including events from all the venues that were reserved for this congregation]

iii. showCongregations
[prints number of Congregations for successful return on the first line followed by a list of Congregation name, type, start and end dates – one per line, -1 for a failure. If no congregations were added, it should simply print a 0]

B. addVenue command now takes the new location string as described earlier "location-address:city:state:postal-code:country". deleteVenue takes an additional argument for country name. Venue names must be unique within a country. There is an additional restriction on deleteVenue now. A venue cannot be deleted if it has any reservation for a currently active or future congregation – meaning deletion of a venue can only occur after all reservations are in the past or deleted. Please ensure that you cleanup any memory allocated for data structures that make up a venue.

i. addVenue <name-string> <location-string> <capacity-integer>
[prints 0 for success, -1 for a failure]

ii. deleteVenue <name-string> <country-string>
[prints 0 for success, -1 for a failure]

C. showVenues command now takes a <location-string> containing minimum 1 and maximum 4 substrings to match city, state, postal-code and country in the format "city:state:postal-code:country". A substring may be empty, e.g. "Boston:MA::USA" indicating all venues in all postal codes of Boston, MA, USA. Country is required in every query. If city is specified then the state must be specified. Alternatively, only postal-code and country may be specified. Thus, you get 4 variations of showVenues: i) country only, e.g. ":::USA"; ii) city, state and country, e.g. "Boston:MA::USA"; iii) postal-code and country, e.g. "::560037:India"; iv) city, state, postal-code and country, e.g. "Bangalore:KA:560001:India". Location address is NOT needed in this query.

showVenues <location-string>
[prints number of matching venues for successful return on the first line followed by a list of venue names, full address and capacity – one per line, -1 for a failure. If no venues match, it should simply print a 0]

D. We can now reserve venues for congregations as follows:

i. reserveVenue <venue-name-string> <venue-country-string> <congregation-name-string>
[prints 0 for success, -1 for a failure. This blocks the venue for the congregation from its start to end date. Note that a venue can be associated with no more than one congregation on a particular date. The congregation and the venue to be reserved must already exist in the system before such a reservation is allowed.]

ii. freeVenue <venue-name-string> <venue-country-string> <congregation-name-string>
[prints 0 for success, -1 for a failure. This frees a previously held reservation of a venue for the named congregation.]

iii. showReserved <congregation-name-string>

[prints number of matching reservations for the specified congregation followed by details of each venue reserved along with full address and capacity – one per line. Prints 0 if no reservations have been made, -1 on a failure.]

E. Each venue must have its own calendar of events showing dates and time in hours and minutes but now an event can only be added under a reservation made by a congregation for that venue. An event can occupy one or more hours on a single date (min 30 mins, max 24 hours). Any event spanning multiple days must be added as multiple events – one for each day. Calendar will not allow multiple events to be scheduled in the same 15-min slot. You should allow reserving calendar slots for events using the following commands:
    a. addEvent <congregation-name-string> <venueName-string> <venueCountry-string> <date-string> <fromTime-string> <toTime-string> <eventName-string>
    [prints 0 for success, -1 for a failure. For this event to be added successfully, the date provided must be within an existing reservation for congregation-name at that venue. Date string takes the format "YYYY-MM-DD" and time string takes the format "hh:mm". Also see constraints on scheduling given above.]
    b. deleteEvent <congregation-name-string> <venueName-string> <venueCountry-string> <date-string> <fromTime-string> <eventName-string>
    [prints 0 for success, -1 for a failure. Event with specified date and exact from time must exist under a reservation by congregation-name to be deleted.]
    c. showEvents <venueName-string> <venueCountry-string> <date-string>
    [prints number of events added on the requested date for successful return on the first line followed by a list of <event name, start time and end time>, one per line; -1 for a failure. If no events were added on the requested date, it should simply print a 0. Note that congregation-name is not needed here.]
    d. showCalendar <congregation-name> <venueName-string> <venueCountry-string>
    [prints total number of events added to the requested venue so far by congregation-name on the first line followed by a list of i) date and number of events on that date (0 is acceptable) on the first line and ii) list (possibly null if no events) of events for that date as <event name, start time and end time>, one per line; -1 for a failure, e.g. if no reservation exists for this congregation at the venue. If no events were added on any date but a reservation exists, it should simply print a 0 (for total events), followed by <date, 0> for each date, one per line. Limit your output to the dates of the congregation.]

Submission Instructions:

Please submit the following to LMS as a single zip file. Separate folders for doc, source and test files:

1. A design document (3-page limit) – see details above.
2. One or more code files (.h or .cpp). You can #include <iostream> to simplify input / output and use C or C++ strings and arrays if you wish. Feel free to use C-style time and date utilities and helper functions from <ctime.h>. Unfortunately, C++11 has

limited facilities to deal with time in <chrono> which is part of std::chrono namespace. Comment your files, C++ classes, functions, other data structures and code as needed. Code must be readable and maintainable by someone other than yourself who may not have access to your design document. You can include scripts you use for testing if you wish.

3. One or more input text files that you use to populate venues and calendars and test your program. Don't just provide inputs for happy path but test for error conditions as well and provide all the input files you have tested your program with. Use descriptive names in the test files to indicate their purpose e.g. populate_venues.txt, wrong_date_error.txt etc.

Please also submit only the source folder to DOMjudge (instructions to follow).