EGC 211 Programming II
C++ Lab 4
2nd September 2024

This assignment will build upon the work you did in Assignment 3 to add some new derived classes and features to your C++ program. This exercise should be done individually – no groups or sharing of code is allowed.

Submission deadline: Sept 11, 11:59 pm

Background: You have now generalized 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. The Product Manager wants to add the following functionality to the next release:

1. So far, the program has treated all congregations uniformly. However, each type has its own specific requirements, namely:
    a. A conference may have a list of individual program or events for which different invitations have to be generated and different rules apply for entry:
        i. One or more workshops – occupying a day or more of the full conference (program type "Workshop")
        ii. The main conference (program type "Main Conference")
        iii. A banquet, perhaps with fancy sit-down dinner (program type "Banquet")
    b. Games will have their own events such as:
        i. Opening ceremony (program type "Ceremony")
        ii. Closing ceremony (program type "Ceremony")
        iii. Sports competitions – let's consider only these program types for now
            1. "Track and field" – includes athletics, football, cricket, hockey as well. Can only be held in an Outdoor Stadium,
            2. "Indoor games" – basketball, volleyball, tennis, badminton etc. Can only be held in an Indoor Stadium.
            3. "Water sports" – swimming and diving. Can only be held in a Swimming Pool.
    c. Concerts will typically have just one venue and:
        i. A pre-concert performance by a lesser-known artist (program type "Pre-concert")
        ii. The main performance by the main artist (program type "Main Concert")
    d. A convention will have everything that a conference has, plus:
        i. Food and beverage stalls selling food and beverages to attendees (program type "Food Court")
        ii. One or more exhibitions on different themes with vendor exhibits (program type "Exhibition")
2. A venue may be one of several types. For example,
    a. An outdoor stadium may allow track-and-field sports, opening and closing ceremonies, concerts and conventions but is likely not appropriate for conferences and may not support indoor sports or swimming

b. A convention center may not allow games but may host conferences and concerts in additional to a convention
c. A concert hall will only support concerts
d. A hotel may support conferences and conventions but not games or concerts

Exercise:

1. Applying object-oriented design methodology, document and add any additional entities and their inter-relationships to your UML diagrams. How will you use inheritance in C++ to model different types of congregations and venues? Limit your class hierarchy to a max depth of 2 inheritance edges (or 3 classes from top to bottom). What would be different in each inherited class from its parent and other sibling classes? Briefly explain your design thinking in a design document (4-5 pages max including diagrams). You are expected to add incremental information to the existing design document where new classes and changes in data structures (if any) are noted.
2. Implement new classes needed and program changes required for the new features in C++. Please see below for changes to the commands used earlier.

A. There is no change to add / delete / show congregations commands but the implementation should now create and initialize objects of more specific classes instead of simply storing the "type" as an attribute of the congregation.
   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 "DD-MM-YYYY". Assume that congregation names are unique across the system, e.g. "Summer Olympics 2024" or "US Open Tennis 2024", "Wimbledon 2025"]
   ii. deleteCongregation <name-string>
   [prints 0 for success, -1 for a failure. This command deletes all associations between the congregation name and]
   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. We will now add at least 2 programs under each Congregation type as follows:
   i. addProgram <congregation-name-string> <program-type-string> <program-name-string> <start-date-string> <end-date-string>
   [prints 0 for success, -1 for failure. Note that a program type should be one of the allowed types under the selected congregation, else it would be an error. Similarly, the named congregation should already exist in the system and the date range being added to the program should fall within the date range of the congregation.]
   ii. deleteProgram <congregation-name-string> <program-name-string>
   [prints 0 for success, -1 for a failure. This command deletes the named program from the congregation. A program cannot be deleted until all of its reservations are deleted first.]
   iii. showPrograms <congregation-name-string>

[prints number of programs added to the named congregation so far on the first line followed by a list of program names, types, start and end dates – one per line, -1 for failure. If no programs were added, it should simply print a 0.]

C. addVenue command now takes an additional argument for type. Venue types to be supported are: Stadiums of 3 types: i) Outdoor Stadium, ii) Indoor Stadium, iii) Swimming Pool; Hotel, Convention Center, Concert Hall. As in assignment 3, name and country are sufficient to uniquely identify a venue.
   i. addVenue <name-string> <location-string> <type-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]

D. showVenues command now takes an optional type-string after location-string to limit the search to venues of specified type (e.g. "Indoor Stadium") if it is provided in the query. If no type is provided, showVenues should continue to work as it did in Assignment 3 but it should list all types of venues. Output now also includes the type of venue.
   showVenues <location-string> [<optional-type-string>]
   [prints number of matching venues for successful return on the first line followed by a list of venue names, full address, type and capacity – one per line, -1 for a failure. If no venues match, it should simply print a 0]

E. Reservations are now made at venues for individual programs of a congregation and not for the whole congregation. To keep things simple, we will enforce these simple rules for reservations:
   i. A stadium, e.g. Outdoor / Indoor / Swimming Pool can be reserved for competition of the appropriate category (Track and field / Indoor Games / Water Sports) respectively. In addition, an outdoor stadium can be reserved for programs in a concert or a convention but not for anything in a conference.
   ii. A hotel can only be reserved for any program in a conference or a convention. You obviously have to specify individual programs in the reservations.
   iii. A convention center can only be reserved for programs in a convention or a concert or a conference but not for any sports competition.
   iv. A concert hall can only be reserved for programs in a concert.
   v. Note that we are not constraining reservations based on capacity at this time.

   Following are the commands for program reservations:
   i. reserveVenue <venue-name-string> <venue-country-string> <congregation-name-string> <program-name-string>
      [prints 0 for success, -1 for a failure. This blocks the venue for the program listed under a congregation from its (program's, not congregation's) start to end date. Note that a venue can be associated with no more than one program on a particular date. Also, a venue can only be reserved if the program type is supported by that venue.]
   ii. freeVenue <venue-name-string> <venue-country-string> <congregation-name-string> <program-name-string>

[prints 0 for success, -1 for a failure. This frees a previously held reservation of a venue.]

iii. showReserved <congregation-name-string>
[prints number of programs added in the congregation followed by program-name and program-type for each program along with the number of matching reservations for that program and then details of each reservation such as venue reserved along with full address and capacity – one per line. Prints 0 if no programs exist, -1 on a failure. If no reservation exists for a program, print program name, type and 0 for number of reservations.]

F. We will skip adding events under reservations in this assignment.

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 (5-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. 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.