**EGC 211 Programming II**

**C++ Lab 1**

**2nd August 2024**

This first lab assignment is designed to refresh your knowledge of C programming that you did in your First Semester. This exercise should be done individually – no groups or sharing of code is allowed. We will build upon the code you write here with C++ knowledge in the next few assignments.

**Submission deadline: Aug 14, 11:59 pm**

**Exercise:**

Design a calendar scheduling system for Summer Olympics (e.g. Paris 2024) in C. The system must have the following minimum features:

1. Support for multiple venues for the games, which should take the following commands from the command line to populate a set of venues in your system:
   1. addVenue <name-string> <location-string> <capacity-integer> [prints 0 for success, -1 for a failure printing the reason for failure on the next line after return value, e.g. out of memory or duplicate venue name]
   2. deleteVenue <name-string>

[prints 0 for success, -1 for a failure printing the reason for failure on the next line after return value, e.g. venue does not exist]

* 1. showVenues

[prints number of venues for successful return on the first line followed by a list of venue names, locations and capacity – one per line, -1 for a failure printing the reason for failure on the next line after return value. If no venues were added, it should simply print a 0]

1. Each venue must have its own calendar showing dates and time in hours. For simplicity, you can assume that the date is a numeric value between 1 and 30 and similarly hour is also a numeric value from 0 to 23. An event can occupy one or more hours on a single date (min 1 hour, 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 hour. You should allow reserving calendar slots for events using the following commands:
   1. addEvent <venueName-string> <date-integer> <fromHour-integer> <toHour-integer> <eventName-string>

[prints 0 for success, -1 for a failure printing the reason for failure on the next line after return value, e.g. out of memory]

* 1. deleteEvent <venueName-string> <date-integer> <fromHour-integer> <eventName-string>

[prints 0 for success, -1 for a failure printing the reason for failure on the next line after return value, e.g. event does not exist]

* 1. showEvents <venueName-string> <date-integer>

[prints number of events added on the requested date for successful return on the first line followed by a list of <event name, start hour and end hour>, one per line; -1 for a failure printing the reason for failure on the next line after return value. If no events were added on the requested date, it should simply print a 0]

* 1. showCalendar <venueName-string>

[prints total number of events added for the requested venue so far on the first line followed by a list of i) date with any event added 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 hour and end hour>, one per line; -1 for a failure printing the reason for failure on the next line after return value. If no events were added on any date, it should simply print a 0 (for total events), followed by <date, 0> for each date, one per line]

**Sample Input and Output:**

Your program should just indefinitely wait for user input when started and terminate on a special input (e.g. “End”) or Ctrl-C / Ctrl-D (end-of-file). You can print a prompt like: “Please enter a command…” initially and after executing each command.

|  |  |
| --- | --- |
| **Input** | **Output** |
| showVenues | 0 |
| addVenue “Stade de France” “Saint-Dennis” 77000 | 0 |
| showVenues | 1  Stade de France Saint-Dennis 77000 |
| addVenue “Roland-Garros” 34000 | -1  Error: provide a string for location |
| addVenue “Roland-Garros” “Paris” 34000 | 0 |
| addVenue “Parc des Princes” “Paris” 48000 | 0 |
| delVenue “Place de la Concorde” | -1  Error: no such venue |
| addEvent “Rolland-Garros” 5 14 16 “Womens Tennis” | 0 |
| delEvent “Rolland-Garros” 5 15 “Womens Tennis” | -1  Error: no matching event (on date 5 starting at hour 15) |
| addEvent “Place de la Concorde” 8 10 12 “Mens Football” | -1  Error: no such venue |
| addEvent “Parc des Princes” 8 10 12 “Mens Football” | 0 |
| showEvents “Rolland-Garros” 6 | 0 (no event on date 6) |
| showEvents “Place de la Concorde” 8 | 1  Mens Basketball 10 12 |
| showCalendar “Rolland-Garros” | 1  1 0  2 0  3 0  4 0  5 1  Womens Tennis 14 16  6 0  …  30 0 |

**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 (1 page limit) describing in your own words about how you are structuring the program and what data structures you are using and why.
2. One or more code files (.h or .c). You can #include libc.h, unistd.h, stdio.h and string.h include files (DO NOT upload these) in your code if you want, but you should not need anything else. Comment your files, C functions, 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.