ENSC 251 FALL 2025 LAB ASSIGNMENT 2

Lab Assignment Overview

In the ENSC 251 course, you will work on four lab assignments using the skills learned throughout this course, i.e., Object-Oriented Programming (OOP) with C++. Each lab assignment is worth 10 marks. All lab assignments are designed to be carried out and evaluated in pairs (i.e., two students per group). You are permitted to work on your own for the assignments (but not the project). This will be more work, and no consideration will be given for the disadvantage you have by working on your own. Groups of three are permitted; however, if you work in a group of three, you will be held to a higher standard.

Please make sure your code can compile and run correctly on the lab computer using onlineGDB. If your code cannot compile, your group can get at most 5 marks. If your code can compile but cannot run, your group can get at most 6 marks. It is highly recommended that you test your code in lab conditions before your demonstration.

You can divide the work roughly based on the listed tasks; the overall marks for major tasks are listed for your guidance, but not the detailed marks for every single item. Please include a simple document in each lab (include a txt or pdf file in the .zip file for submission) indicating which parts each member finished. There is some negative marking if you don't complete the listed tasks; details are not listed. Basically, the detailed grading scheme except the overall marks for major tasks for each lab will not be released before your lab grading is done. Think about being in a real interview; nobody will tell you what the detailed grading schemes are. I also reserve the right to adjust grades based on a holistic evaluation of your submission.

Lab Assignment 2

In the first two lab assignments, you will build a simple booking system for a warehouse, where one can reserve, query, and release storage time slots for loading and unloading goods. To make it simpler, we make the following assumptions: *(unchanged from Assignment 1)*

1. The storage periods supported in the system are from Week 1 (starting Jan 1, 2025) to Week 52 (ending Dec 31, 2025), i.e., the full year of 2025.
2. No reservations can be made during maintenance weeks or peak seasons.
3. The slot format for each week is in hourly increments, from Slot 0 (midnight to 1 AM) to Slot 23 (11 PM to midnight). We only care about whole hours and ignore minutes.
4. A reservation in the booking system can only be made in multiples of 2 hours, and it only supports same-week reservations. For example, Week 5 Slot 9 to Slot 10 (9 AM to 11 AM) is valid, Week 5 Slot 10 to Slot 13 is also valid, Week 5 Slot 10 to Slot 12 is invalid because it's not a multiple of 2 hours, Week 4 Slot 20 to Week 5 Slot 2 is invalid because it crosses weeks.

Extend the system to support creating, querying, and canceling reservations. Introduce a new **Booking** class to represent individual reservations, and a **BookingSystem** class to manage them across the year.

1. [2.5 marks] Implement the Booking class as an ADT.

- Private members: int bookingID (unique auto-incrementing ID starting from 1), Week week, SlotRange range, string customerName.
- Constructor: Take week number, beginSlot, endSlot, customerName; validate using Week/SlotRange isValid; assign next ID.
- Accessors/mutators: Get/set for all members (setters should re-validate).
- isConflicting(const Booking& other): Check if this booking overlaps with another (same week and overlapping slots).
- Output function: Print booking details (ID, week, slots with times, customer), using ostream&.

2. [3.5 marks] Implement the BookingSystem class as an ADT.

It manages all bookings and availability:

- Private members: Booking bookings[200]; int bookingCount (to track number of bookings); Week weeks[52] (as in Lab 1, initialized 1–52 with reserved false; set maintenance weeks).
- Constructor: Initialize weeks array (define global const peakWeeks, e.g., 22–24, 35–37, 51–52).
- canReserve(int weekNum, SlotRange range): Validate week/range; check not maintenance/peak; scan bookings array for conflicts in that week; check reserved slots are free.
- addBooking(int weekNum, SlotRange range, string customerName): If canReserve true, create Booking, add to bookings array, increment bookingCount, mark reserved[] true in the Week. Return bookingID or -1 on failure (print error). If array is full (bookingCount >= 200), return -1 (print error).
- queryBooking(int bookingID): Find and print the Booking details in the bookings array; return true if found.
- cancelBooking(int bookingID): Find Booking in array, mark as inactive (e.g., set bookingID to -1 or use a flag), unmark reserved[] in Week. Return true on success (print confirmation/error).
- checkWeek(int weekNum): Print the free/reserved slots for the specified week.

3. [0.5 mark] Update Week class:

- Add markReserved(SlotRange range, bool status) to set/unset multiple reserved slots at once (used in add/cancel).

4. **Declare/define** Booking in booking.hpp/cpp; BookingSystem in bookingsystem.hpp/cpp.
5. **[1.5] Create a main.cpp loop and implement commands.**
   - Create BookingSystem object.
   - Loop: Read commands ("reserve ", "query ", "cancel ", "check ", "exit").
   - Process commands all parameters are integers:
     1. reserve [week] [slotStart] [slotEnd] (reserve a slot and print ID)
     2. query [bookingID] (print booking details of given ID)
     3. cancel [bookingID] (remove the booking ID and print message indicting success)
     4. check [bookingID] (print free/reserved slots for the week).
     5. Exit. (exit program)
   - After each reserve/cancel/query command, print the week's free/reserved slots.
   - Handle invalid inputs (e.g., overlaps, invalid weeks) without crashing.
6. **[2.0 marks] Testing/demonstration:**
   - Use the loop commands to test and demonstrate the code.

Provide strong comments, naming, and style; these impact marks.

Assignment Submission

Submit lab2.zip via Canvas with all .hpp/.cpp files. Due 11:59 PM Sunday, October 26, 2025.

Lab Demonstration

Sign up for ~8-minute TA demo in lab sessions. Use submitted code only; explain with menu examples.