CS201 – SPRING 2020-2021
Take-Home Exam 5
– Boat Reservation Database with Classes –

Due June 1 Tuesday, 23:55 (gets 10 points bonus)

LATE Due June 6 Sunday, 23:55 (no penalty)

Introduction

The aim of this take-home exam is mainly to practice on classes, as well as vectors and file streams.

Your take-home exams will be automatically graded using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check whether your implementation is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the take-home exam (**all** .cpp, .h and .txt files for this take-home exam). Additionally, you should submit all your .cpp and .h files to SUCourse **without zipping** them. **Just a reminder, you will see a character ¶ which refers to a newline in your expected output.**

The name of your main source (cpp) file should be in the expected format: "SUCourseUsername_THEnumber.cpp" (all lowercase letters). Please check the submission procedures of the take-home exam, which are listed at the end of this document.

To get help using GradeChecker you may ask questions to the list of your grader TAs: cs201gchelp@lists.sabanciuniv.edu.

Description

Your program will first start by reading a filename from the console, which will be the reservations file for boat reservations. The other two filenames "Boats.txt" and "Sailors.txt" must be hardcoded (i.e. you must define them in your program). If any of the files cannot be opened, just display the message "`Cannot open the files. Exiting…`" and exit from the program.

The columns in these files are separated by tab characters, and rows are separated by newline characters. First rows of the Boats.txt and Sailors.txt files are the column identifier. The Reservation file does not have a column identifier. You will read all of the three files.

A sample line from the boat database is as follows:

```
101  INTERLAKE BLUE
```

Here, 101 is the boat id, INTERLAKE is the boat's name and BLUE is the boat's color.

A sample line from the sailor database is as follows:

```
22   Dustin   7.1 45.0
```

Here, 22 is the sailor id, Dustin is the sailor's name, 7.1 is his rank and 45.0 is his age.

So far it is the same as the previous take home exam but for the reservations database, we want you to create a class named **Reservations** and add and the user the following class member functions for the actions defined below:

1) Add a Reservation

For the Add operation, the line in the reservations file starts with "A", followed by the sailor id, the boat id and the reservation date. An example line is as follows:

```
A    64   101  2014-02-18
```

For this operation, you should implement and use this add member function in the **Reservations** class. The **Reservations** class has a private vector of ***Reservation*** instances to represent the database.

2) Delete a Reservation

For the Delete operation, the line in the reservations file starts with "D", followed by the sailor id, the boat id and the reservation date. But be aware that sid, bid or reservation date can be 0, meaning that you will implement three versions of this Delete function:

1. If the sailor id is 0, (and date is 0000-00-00) this means you should delete all records of the boat with the given boat id.
2. If the boat id is 0, (and date is 0000-00-00) this means you should delete all records of the sailor with the given sailor id.
3. Otherwise, all information of the reservation is provided (sailor id, boat id and reservation date) so you are required to delete a specific record from the database.

If the delete operation fails for any of these cases, you should display an error message saying starting with "`Error: Could not delete reservation with ...`" appending the boat id, sailor id and/or reservation date information to it (see sample runs).

3) Find a Reservation

For the find operation, the line in the reservations file starts with “F”, followed by sailor id, boat id and reservation date(s).

1. If both sailor id and boat id are 0, then two dates are provided (start date, end date) and you should find all reservations between these dates. Start and end dates are included.
2. If only sailor id is 0, then you should find the reservation for the provided boat id, on the provided date.
3. If only the boat id is 0, then you should find the reservations for the provided sailor id, on the provided date.

After you find the reservation(s), you should print them to the console. If no reservations are found for this query, you should print an error message starting with “`Error: No matching reservation found with ...`” appending the boat id, sailor id and/or reservation date(s) information to it (see sample runs).

The Classes

In this homework, you are **required** to implement the necessary classes. If you do not use classes in your homework solution, you will not get any points for the sample runs, even if they work correctly.

1) Reservations

`Reservations` class is where you implement all member functions for add, delete and find operations. Explained below:

- ***AddReservation*** function, which takes a **`Boat`**, a **`Sailor`** and a **`Date`** variable and adds this reservation to the database. After each add operation, you should sort the database according to the reservation **`Date`**’s, and if the dates are the same then you should sort the reservations according to the sailor’s name.

- ***DeleteReservations*** function should be implemented in three forms to suit the definition of delete operation explained above. After each delete operation, the database should be kept sorted and there should be no blank records in the database.
  First form accepts a single Boat variable, and deletes all records for this boat.
  Second form accepts a single Sailor variable, and deletes all records for this sailor.
  Third form accepts a Date, a Sailor and a Boat variable and deletes a specific reservation matching these variables.

All three functions must return true if the delete operation is successful, and false otherwise.

- ***FindReservations*** function. This function should return a vector of Reservation (below class) and it also has three different versions:
    - First version accepts two Date class instances, a start date and an end date. You should return a vector of all reservations between(and including) the start and end dates.
    - Second version accepts a Boat and a Date variable, and returns all reservations for this boat on this Date.
    - Third version accepts a Sailor and a Date variable, and returns all reservations for this sailor on this Date.

2) Reservation

You will use this **`Reservation`** class to hold the reservation records in one place. Reservation class will have private variables Sailor, Boat and Reservation Date. You should write a constructor accordingly. You'll be given the default constructor in the Reservations.h file.

You also need a *Print* function so that you can display the Reservation information. Output of the print function is of the following format:

```
February 19 2014 -> Dustin(45,7.1) has reserved INTERLAKE(BLUE)
```

3) Sailor

This **`Sailor`** class is very similar to the Sailor struct in the previous take home exam. The variables are the same, and you'll need a constructor to create an instance of this class.

4) Boat

This **`Boat`** class is very similar to the Boat struct in the previous take home exam. The variables are the same, and you'll need a constructor to create an instance of this class.

You are free to implement get/set functions or extra constructors as needed for all classes.

Please refer to the "Sample Runs" section for some examples and further details.

IMPORTANT!

If your code does not compile, then you will get **zero**. Please be careful about this and double check your code before submission.

VERY IMPORTANT!

Your programs will be compiled, executed and evaluated automatically; therefore you should definitely follow the rules for prompts, inputs and outputs. See **Sample Runs** section for some examples.

- **Order of inputs and outputs** must be in the mentioned format.

Following these rules is crucial for grading, otherwise our software will not be able to process your outputs and you will lose some points in the best scenario.

No abrupt program termination please!

Especially during the input check, you may want to stop the execution of the program at a specific place in the program. Although there are ways of doing this in C++, it is not a good programming practice to abruptly stop the execution in the middle of the program. Therefore, your program flow should continue until the end of the main function and finish there.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You should follow the input order in these examples and the prompts that your program will display **must** be **exactly the same** as given in the following examples.

Sample Run 1

```
Enter filename for reservation database: Reservations.txt

Cannot open the files. Exiting...
```

Sample Run 2

```
Enter filename for reservation database: **Reservations1.txt**
Error: Could not delete reservation "31 102 2014-02-28"

Error: Could not delete reservation for sailor id 50

Error: Could not delete reservation for boat id 105

Find Results:
February 19 2014 -> Dustin(45,7.1) has reserved INTERLAKE(BLUE)

Find Results:
March 14 2014 -> Brutus(33,1.1) has reserved INTERLAKE(RED)

Find Results:
February 18 2014 -> Horatio(35,7) has reserved INTERLAKE(BLUE)
February 19 2014 -> Dustin(45,7.1) has reserved INTERLAKE(BLUE)
March 3 2014 -> Brutus(33,1.1) has reserved MARINE(RED)
March 10 2014 -> Andy(25.5,8) has reserved CLIPPER(GREEN)
```

Sample Run 3

```
Enter filename for reservation database: **Reservations2.txt**
Find Results:
Error: No matching reservation found for boat id 101 on 2014-02-19

Find Results:
March 14 2014 -> Brutus(33,1.1) has reserved INTERLAKE(RED)

Find Results:
February 18 2014 -> Horatio(35,7) has reserved INTERLAKE(BLUE)
February 27 2014 -> Rusty(35,10) has reserved TITANIC(BLACK)
February 28 2014 -> Lubber(55,8) has reserved INTERLAKE(RED)
March 12 2014 -> Art(25.5,3) has reserved INTERLAKE(BLUE)
March 14 2014 -> Brutus(33,1.1) has reserved INTERLAKE(RED)

Find Results:
Error: No matching reservation found for sailor id 29 on 2014-03-03

Find Results:
Error: No matching reservation found between dates 2014-05-01 & 2014-
05-14
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all take-home exams, unless otherwise noted.

How to get help?

You can use GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

You may ask questions to TAs (Teaching Assistants) or LAs (Learning Assistants) of CS201. Office hours of TAs/LAs are at the course website.

What and Where to Submit

You should prepare (or at least test) your program using MS Visual Studio 2012 C++ (Windows users) or using Xcode (macOS users).

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your name and last name is "Barış Altop", and if you want to write it as comment; then you must type it as follows: *// Baris Altop*

Submission guidelines are below. Since the grading process will be automatic, students are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be 0.

- Name your submission file as follows:
  - Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
  - Name your cpp file that contains your program as follows: "**SUCourseUsername_THEnumber.cpp**"
  - Your SUCourse user name is actually your SUNet username, which is used for checking sabanciuniv emails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name (**use only lowercase letters**). For example, if your SUCourse username is "**altop**", then the file name should be: **altop_the1.*cpp*** (please only use lowercase letters).
  - Do not add any other character or phrase to the file name.
- Please make sure that this file is the latest version of your take-home exam program.
- Submit your work **through SUCourse only**! You can use GradeChecker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse. You will receive no credits if you submit by any other means (email, paper, etc.).

- If you would like to resubmit your work, you should first remove the existing file(s). This step is very important. If you do not delete the old file(s), we will receive both files and the old one may be graded.

Grading, Review and Objections

Be careful about the automatic grading: Your programs will be graded using an automated system. Therefore, you should follow the guidelines on the input and output order. Moreover, you should also use the same text as given in the "Sample Runs" section. Otherwise, the automated grading process will fail for your take-home exam, and you may get a zero, or in the best scenario, you will lose points.

Grading:

- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
- Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the Syllabus or on the course website.

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your take-home exam from the email address provided in the comment section of your announced take-home exam grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the take-home exam tab to see the problem with your take-home exam.
- Download the file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

Good Luck!
Şevval Şimşek & Gülşen Demiröz & Barış Altop