Name: Otto Häger

Date: 3/18/24

CSCI 1300: Recitation 9

Spring 2024

Please make sure to write your name and the date in the top left corner. You may use any course materials to answer the following questions and you may collaborate with others, but the work shown must be your own.

1 The Booking Struct

You are running a Hostel, with multiple shared rooms and each room has exactly 10 beds. You want to keep track of the bookings made for the rooms in your hostel. To manage this you begin with creating a `Booking` struct and a function `displayBooking` to print out all the bookings.

Note : Write a main() function that creates an array of Booking objects. Call `displayBooking` to print the details of the booking. You may use bookings1.txt file on Github for example data.

Member Type	Member Name	Description
string	`name`	Name of the person who made the booking
int	`num_people`	Number of people in the booking

Function: displayBooking(Booking[], int)	`void displayBooking(Booking bookings[], int num_bookings)`
Purpose:	Display all the bookings stored in the bookings array.
Parameters:	**Booking bookings**: An Array containing Booking objects **int num_bookings**: The size of bookings[] array
Return value:	The function doesn't return any value.
Error handling/ Boundary conditions:	If num_bookings <=0, exit the function.
Example:	**Sample Code 1:** (see below)

Sample Code 1:

```
// Assume the proper libraries are included
// Assume the proper implementation of the struct
↪  is included

int main()
{
    Booking booking1;
    booking1.name = "Jacob";
    booking1.num_people = 4;
    Booking booking2;
    booking2.name = "Suzy";
    booking2.num_people = 5;
    Booking booking3;
    booking3.name = "Alex";
    booking3.num_people = 1;

    int num_size = 3;
    Booking bookings[3] = {booking1, booking2,
     ↪  booking3};
    displayBooking(bookings, num_size);
}
```

Name:

Date:

CSCI 1300: Recitation 9

Spring 2024

Example (continued):	**Sample Output 1.1:**
	
	 `Jacob 3`
	 `Suzy 5`
	 `Alex 1`

Problem 1.1. Write out the steps you would use to solve this problem by hand as pseudocode.

```
iterate through array of bookings from 0 to
array size - 1.
    Print Object name and Object number
```

Problem 1.2. Pick two possible inputs for your program. These can be in the form shown in the sample run above, or as lines of a booking file. Try to pick values that will test different aspects of your function. Follow the steps you wrote for these values to find your result, and verify it.

Alex, 3 → Alex 3

Bengt, -1 → [no OutPut]

Problem 1.3. Implement your solution in C++ using VS Code. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do. Make sure you test for the values used in your sample runs, and for the boundary conditions.

Name: Spring 2024

Date:

CSCI 1300: Recitation 9

2 The Room Class

Now lets create a Room class as described below:

Data Members (private):

Member Type	Member Name	Description
string	`_room_id`	ID of the room
const int	`_MAX_OCCUPANTS`	Maximum number of occupants that can be added to the room, set as 10
int	`_num_current_occupants`	Number of people currently occupying the room
int	`_num_current_bookings`	Number of bookings currently added to the room
Booking[]	`_bookings`	Array containing all the bookings currently added to the room. The size of this array is `_MAX_OCCUPANTS`

Member Functions (public):

Function	Description
Default constructor	Creates a new instance of Room by setting `_room_id` to an empty string, `_num_current_occupants` to 0 and `_num_current_bookings` to 0
`Room(string)`	Creates a new instance of Room by initializing `_room_id` with the string parameter, `_num_current_occupants` to 0 and `_num_current_bookings` to 0
`void setRoomId(string)`	Sets the `_room_id` to the value of the string parameter
`string getRoomId()`	Returns the `_room_id` of the Room
`void setNumCurrentOccupants(int)`	Sets the `_num_current_occupants` to the value of the integer parameter
`int getNumCurrentOccupants()`	Returns the value of `_num_current_occupants`
`bool addBooking(string)`	Takes a string (the name of the file to be read) and populates the `_bookings` array with Booking objects. Returns a true if all the bookings in the file were successfully added and false otherwise. *See Function Specification table below for more details*
`bool removeBooking(string)`	Takes a string (the booking name) and removes the booking from the array `_bookings`. Returns a true if removed successfully and false otherwise. *See Function Specification table below for more details*

Name:

Date:

Spring 2024

CSCI 1300: Recitation 9

Problem 2.1. Create the above class definition in Room.h. Add the Booking struct from above in Room.h. You have to follow the table above during this process.

Problem 2.2. Implement all the member functions in Room.cpp. The function specifications for some of the member functions are given below.

Function: bool addBooking(string)	`bool addBooking(string filename)`
Purpose:	Reads the file and populates the `_bookings` array with Booking objects.
Parameters:	**filename**: A file containing the booking details
Return value:	Returns true if all the bookings in the file were successfully added to the room, otherwise returns false.
Error handling/ Boundary conditions:	Return false if the room already has `_MAX_OCCUPANTS`. If all the bookings in a file cannot be added to the room, display the current booking that cannot be added, along with the remaining available space in the room and return false.

Name:

Date:

CSCI 1300: Recitation 9

Example:	**Sample Code 2:** (see below)
Example (continued):	**Sample Output 2.1:** (see below)

Sample Code 2:

```
// Assume the proper libraries are included
// Assume the proper implementation of the class is
↪  included

int main()
{
    //create a Room object
    Room room("1B25");

    //test return value for a file that doesn't
    ↪  exist
    assert(room.addBooking("fake_file.txt") == 0);

    //test return value for a file that exist and
    ↪  all booking in the file CANNOT be
    ↪  accomodated in the room
    assert(room.addBooking("bookings1.txt") == 0);

}
```

Sample Output 2.1:

```
The file cannot be opened!

Cannot accommodate the booking for 'Dave,5' in the
room 1B25!!  You must book in smaller parties!
The room can accommodate only 4 more!
```

Sample Code 3:

```
//create a Room object
Room room("1B25");

//test return value for a file that exist and all booking in the file CAN be
↪  accomodated in the room
assert(room.addBooking("bookings2.txt") == 1);

//test return value for a file when the room is already at capacity
assert(room.addBooking("bookings3.txt") == 0);
```

Sample Output 2.2:

```
Successfully added all bookings.

Sorry Room 1B25 is already at capacity!!
```

Name: Spring 2024

Date:

CSCI 1300: Recitation 9

Notes:

- Empty lines should not be added to the array.
- You may assume that each non-empty line will contain valid data, i.e. each non-empty line will consist of a Booking name, and the number of people in the booking, separated by a comma.
- There will be exactly one comma on each line.
- Booking names may have spaces in them.
- Assume that the txt files provided do not have duplicate booking listed.
- You should use the `stoi()` method to convert the number of people in each booking from a string to an int.
- Update the `_num_current_occupants` and `_num_current_bookings` data members appropriately.

Function: bool removeBooking(string)	`bool removeBooking(string booking_name)`
Purpose:	Removes the specified booking from $_bookingsarray$.
Parameters:	**`booking_name`**: The booking name of the booking to be removed.
Return value:	If the booking is removed successfully, displays the updated Room stats and return true otherwise returns false.
Error handling/ Boundary conditions:	Return false if the room is already empty. Return false if the booking name is not found in the room .
Example:	**Sample Code 4:** `// Assume the proper libraries are included` `// Assume the proper implementation of the class is included` `int main()` `{` `//create a Room object` `Room room("1B25");` `//test return value for a file that exist and all booking in the file CAN be accomodated in the room` `assert(room.addBooking("bookings2.txt") == 1);` `//test return value for removing a booking which is NOT added to the Room` `assert(room.removeBooking("Sara") == 0);` `//test return value for removing a booking which is already added to the Room` `assert(room.removeBooking("Leo") == 1);` `}`

Sample Code 4:

```
// Assume the proper libraries are included
// Assume the proper implementation of the class is
↪  included

int main()
{
    //create a Room object
    Room room("1B25");

    //test return value for a file that exist and
    ↪  all booking in the file CAN be accomodated
    ↪  in the room
    assert(room.addBooking("bookings2.txt") == 1);

    //test return value for removing a booking
    ↪  which is NOT added to the Room
    assert(room.removeBooking("Sara") == 0);

    //test return value for removing a booking
    ↪  which is already added to the Room
    assert(room.removeBooking("Leo") == 1);

}
```

Name:

Date:

CSCI 1300: Recitation 9

Spring 2024

Example (continued):	
	Sample Output 2.3:

Sample Output 2.3:

```
Successfully added all bookings.

Booking Sara not found in room 1B25

Removed booking:  Leo from room 1B25
Updated Room stats for room:  1B25
Number of Current Booking:  3
Number of Current Occupants:  7
```

Sample Code 5:

```
//create a Room object
Room room("1B25");

//test return value for removing a booking when the Room is empty
assert(room.removeBooking("Sara") == 0);
```

Sample Output 2.4:

```
There are no occupants to remove in 1B25
```

Notes:

- Update the `_num_current_occupants` and `_num_current_bookings` data members appropriately.
- The booking name should be case insensitive.

Problem 2.3. Write a main() function that creates Room objects and tests all the constructors and member functions of Room class.

Submission Instructions: Create a zip file that contains your solution .cpp file for question 1, question 2, and photos of this handout and submit on Canvas.