**CIS 2275 C++ Programming Part 2**

**Hotel Reservations and Date classes**

**Turn in requirements:**

1. Please name your Project LastnameP5, such as NelsonP5.
2. Upload your project to Visual Studio Online. As backup, remove the .sdf, Browse.VC.db and 2 debug/x64 files and then upload your zipped project through BlackBoard.

**Program Requirements:**

1. **3 pts** Write your name, email address and file name at the top of your source code in a comment.
2. **5 pts** Use good C++ programming style and formatting for your program.
3. **5 pts** Use appropriate comments to explain what you are doing. Be sure that your program output is neatly presented to the user and there are no spelling or grammatical errors.

In this program we are going to write a C++ class called HotelRes that makes use of the Date class from our book. Your HotelRes class is used by the C++ Spa and Resort for its guest

reservations. The HotelRes object will handle the user’s information regarding a stay at the Spa. It also reads the current room and tax rates from a file, determines the total cost of the stay, and writes a Reservation Confirmation file for the user.

The reservation form that will be used to capture the data will be programmed on a Windows form and should look something like this:



Put your name, program number and title in the Form1 Text property. Add a label or textbox, txtMenu, todisplay the program description, room rates and discount. Use a textbox for the user to enter the reservation name. Add a label to give instructions. Use a numerical up down (nud), with label, to find out how many in the party.

Have a label or text box to display the reservation confirmation. Use nuds to allow the user to select the month, day and year of arrival and departure for the stay. You can do a lot of input validation by choosing good endpoints for the nuds. We will allow the user to make a reservation for one year past the booking year, so the year will be either 2019 or 2020.

You will need two buttons, one to make the reservation, to be pushed only when all the data has been input, and one to clear the screen so that another reservation can be made. You will also need to create a MyForm\_Load event so that you can instantiate a HotelRes object using the default constructor, then display the room menu in the appropriate label or textbox before the user can enter any data. That object is local to the Form\_Load event and will disappear when the form has completed loading.

I will show you how to handle strings and output from nuds in class, strings will require the UtilityFunction.h file.

The interface between the form and the classes will be in the event handler for the Make Reservation button. A HotelRes object will be created using the overloaded constructor for each reservation made. Both constructors of this class call the ReadReservation private method, which reads the Reservation data file (name it ResData.txt). This file contains the room rate based on one night’s stay, lodging tax rate in %, discount rate for staying 5 days or more. You may assume each reservation is for 1-2 adults, non-smoking room, 1 room only (same price for each person).

The file should look like this:

185.00

12.33

15.00

The Hotel Reservation class should follow this format:

class HotelRes

{

private:

string name; //name of reservation party

int guests; //number of guests in a room, 1 or 2

Date arrival, departure, booking; //three date objects for guest data

int numberOfNights; //length of stay at the spa

double roomRate; //basic rate of room, not including potential discount

double lodgingTax; //tax rate percentage based on the city/county/state laws

double discountRate; //percentage discount for staying 5 nights or more

string confirmationNumber; //confirmation number based on 10001+lastname

string errorstring; //string reporting validation error

double roomCost, tax, totalCost;

bool bReadIn, bWrittenOut; //shows input/output files read/written status

string reservationDescription;

static int resNumber; //the number of the reservation made, declaration only

void ReadReservationData(); //reads file with hotel reservation data

void CalculateCostOfVisit(); //calculates total cost to guest(s)

void WriteConfirmationFile(); //writes the complete confirmation file

void ValidateReservation(); //validates arrival/departure dates

void MakeReservation();

public:

HotelRes(); //default constructor—calls ReadReservationData

HotelRes(string resName, int guests, Date &arr, Date &dpt, Date &book);

void SetName(string n){name = n;} //reservation name

void SetDates(Date &arr, Date &dpt, Date& book);

void SetNumberOfGuests(int g){guests = g;} //number of guests in a room

string PresentRoomMenu();

string GetReservationDescription(){return reservationDescription;}

};

**You may modify this class declaration to fit your model. Please don’t be extreme. You know who you are.**

The HotelRes’s MakeReservation function is called from the overloaded constructor and from the SetDates method. In turn, MakeReservation first calls ValidateReservation to make sure the data is valid and that the user doesn’t enter dates that don’t make sense—such as departing before they arrive or arrival before the booking date. If the date is not valid, errorString is set to report the invalid date and reason.

There is another aspect of validating dates, and that is whether the data input is valid. For example, someone might mistakenly enter 2/31/ 2020. Whose job is this aspect of validation? Should it be the Date class or the HotelRes class?

We will add a method to the Date class, bool ValidateThisDate() to perform the validation.

If the Date objects have been validated, MakeReservation calls CalculateCostOfVisit and WriteConfirmationFile. It also increments resNumber.

The cost can be determined this way: Each night is the basic room rate (here is it $185.00).

If one user were staying two nights:

Total Nights: 2 2 x $185 = $370.00 room cost

Lodger’s tax 12.33% $370 x 0.1233 = $45.62

Total Cost $415.62

If two persons were staying 5 nights or longer, they receive a 15.0% discount off the room cost

Total Nights: 6 6 x $185 = $1110.00 less 15.0% = $943.50 room cost

Lodger’s tax 12.33% $943.50 x 0.1233 = $116.33

Total Cost for one person $1059.83

Total Cost for two people $1059.83 x 2 = $2119.66

The WriteConfirmationFile opens a file that reports all of the reservation data. The filename is composed of the ConfirmationNumber + “.txt” The confirmation numbers begin at 10001 and the reservation confirmation number is the 10000 number with the first 2 letters of the guest’s last name in upper case. For example, if the first guest were John Brown, the confirmation number would be 10001BR. The second guest Bob Smith, the confirmation number would be 10002SM. If the third customer were Nelson, the confirmation file would be: 10003NE.txt. It might look like this:

*Welcome to the C++ Spa and Resort August 17, 2015*

*Guest: Elmer Fudd Confirmation number = 10001FU*

*Arrival: December 1, 2015 Departure: December 9, 2015*

*Number of nights: 8 Number of Guests: 2*

*Room Cost: (with 15.00% discount) $2516.00*

*Lodging tax: $310.22*

*TotalCost: $2826.22*

*We look forward to seeing you in December!*

Your confirmation file must have a month-specific comment in it, such as “We look forward to seeing you in March.”

**Static Class Variables:** For the static class member variable ( static int resNumber;) , the variable definition must be made the class implementation file, otherwise known as HotelRes.cpp. At the same time, it can be initialized. This statement might look like:

int HotelRes::resNumber = 0;

This statement is the actual definition of the static variable and allocates the memory for it.

The purpose of this variable is to keep track of the number of reservation objects that are created.

In the Date class, we need to use the monName[12] array in more than one method. It is cumbersome to re-declare this array in each method. It can be declared static also, and accessed in any class method. Since it is fixed, it can also be declared const.

**Constructor Initializer:** We will also use a constructor initializer to initialize the date objects through the HotelRes overloaded constructor. First, get the user data, including the arrival and departure dates. Create Date objects to hold the date data. Finally, use the overloaded constructor to create the HotelRes object.