**Mini Project 03 Reservations**

**Due Thursday, March 7, 11:59 PM**

In this mini project, you are asked to process different types of reservations and determine the cost of each. The starter project contains multiple files.

## The Date Class

The ***"Date.h"*** file has started the definition of a Date class. It contains only one object-level data member, days_, that stores the number of days that have passed since the beginning of the year. That is, 0 refers to January 1, 1 refers to January 2, and so on. The default constructor has been implemented to set the days_ data member to 0, representing January

1.
```
class Date {
  private:
    unsigned days_;
  public:
    Date();
};
```

You are to add a class-level constant array of 12 elements where each element holds the number of total days in the corresponding month. You may assume February has 28 days.

In addition, add the prototypes of the following functions to the **"Date.h"** file, implement them in the **"Date.cpp"** file, and demonstrate thorough testing in the **"main.cpp**" file. It's important that you ensure the Date class has been properly implemented and tested **BEFORE** working on the different Reservation classes.

- The overloaded << operator as a friend function to support the display of a Date object. This function is to send a message about the Date object to an output stream. The first part of the message must be a date in the "MM/DD" format. It is then followed by the number of days for this date from January 1. **Hint:** Use the class-level constant array to help determine the corresponding month and day values for days_.

For example, given the following code segment:

```
Date d;
cout << d << endl;
```

Here's its expected output:

```
The date 01/01 is 0 day(s) from January 1
```

- The overloaded = operator as a member function to support the assignment from a string on the right to a Date object on the left. You may assume that the string on the right of = will always represent a valid date in a non-leap year using the "M/D" format. The date may have one or two digits in the month and year. For example, March 7th would be represented as

"3/7" and not "03/07". The function is to set days_ member of the invoking Date object to the number of days for the given date from January 1 and to return a reference to the invoking object. **Hint:** Use the class-level constant array to help determine the value for days_.

For example, given the following code segment:

```
d = "2/14";
cout << d << endl;
```

Here's its expected output:

```
The date 02/14 is 44 day(s) from January 1
```

- The overloaded >= operator as a member function to support the comparison between a Date object on the left and a string on the right in the format "M/D". Again, the date may have one or two digits in the month and year. The function shall return true if the Date object on the left of >= has a later date than the string object on the right. Otherwise, false shall be returned. For example, given the following code segment:

```
today = "9/26";
cout << boolalpha;
cout << "today >= 9/10 returns " << (today >= "9/10") << endl;
cout << "today >= 9/30 returns " << (today >= "9/30") << endl;
```

Here's its expected output:

```
today >= 9/10 returns true
today >= 9/30 returns false
```

- The overloaded <= operator as a member function to support the comparison between a Date object on the left and a string on the right in the format "M/D". Again, the date may have one or two digits in the month and year. The function shall return true if the Date object on the left of <= has an earlier date than the string object on the right. Otherwise, false shall be returned. For example, given the following code segment:

```
Date today;

today = "9/26";
cout << "today <= 9/10 returns " << (today <= "9/10") << endl;
cout << "today <= 9/30 returns " << (today <= "9/30") << endl;
```

Here's its expected output:

```
today <= 9/10 returns false
today <= 9/30 returns true
```

## The Reservation Class

This class is created to handle generic reservations or bookings. Add the ***"Reservation.h"*** and "***Reservation.cpp***" files to define and implement the following Reservation class. Add appropriate statements in the main function to perform thorough testing of related member functions in the "***main.cpp***" file.

```
class Reservation {
private:
  string reservationName_; // reservation name
  Date reservationDate_;   // reservation date
  float reservationFee_;   // reservation fee
protected:
  bool is_high_season() const;

public:
  Reservation(const string &reservationName, int day, int month, float fee = 0);
  double calculate_cost() const;
  string to_string() const;
};
```

Here is more information about the member functions of the Reservation class.

- `Reservation(const string &reservationName, int day, int month, float fee = 0);`

  This function is to create a Reservation object by setting the data members using the parameters provided.

- `string to_string() const;`

  This function is to return a string explaining the invoking Reservation object. For example, given the following code segment:

  ```
  Reservation r1("Valentine Day Reservation", 14, 2, 20);
  cout << "\nReservation Details:" << endl;
  cout << r1.to_string() << endl;
  ```

  Here's its expected output:

  ```
  Reservation Details:
  Valentine Day Reservation on 02/14
  ```

  Hint: reservation date can be extracted using the << and then >> operators. First, the << operator overloaded for the Date class can be used to send the following into a stringstream object since stringstream is a derived class of the ostream class.

  ```
  The date 02/14 is 44 day(s) from January 1
  ```

After that, you can use the >> operator to extract the “02/14” part of the above stringstream object into a string variable.

- `bool is_high_season() const;`

  This function is to return true if the reservationDate_ data member of the invoking object falls in the high season. The high season is defined as any date between December 20 and January 6. To help you test this function, you may temporarily move it from protected into public. It’s important to make sure this function is working properly before working on the calculate_cost member function.

- `double calculate_cost() const;`

  This function is to first call the is_high_season function to verify whether the reservation is during the high season or not. If the reservation is in the normal season, the function shall return the reservationFee_ data member. If the reservation is in the high season, the function shall return the reservationFee_ multiplied by 2.

## The HotelReservation Class

This class is defined to handle hotel reservations. Add the ***“HotelReservation.h”*** and “***HotelReservation.cpp***” files to define and implement the following derived class of the Reservation class. Add appropriate statements to perform thorough testing of related member functions in the “***main.cpp***” file.

```
class HotelReservation : public Reservation {
private:
  string hotelName_;    // hotel name
  double costPerNight_; // cost per night
  int nights_;          // number of nights

public:
  HotelReservation(const string &reservationName, int day, int month,
                   const string &hotelName, double costPerNight, int nights);
  double calculate_cost() const;
  string to_string() const;
};
```

- `HotelReservation(const string &reservationName, int day, int month, const string &hotelName, double costPerNight, int nights);`

  This constructor is to create a new HotelReservation object using the parameters provided.

- `string to_string() const;`

  This function is to override the to_string function in the base class and return a string explaining the invoking HotelReservation object.

- `double calculate_cost() const;`

  This function is to override the calculate_cost function in the base class and return the reservation cost as follows: If the reservation is made during the high season, then the function shall return the product of the cost per night, the number of nights, and a surcharge of 50%. Otherwise, the function shall return the product of the cost per night and the number of nights.

  For example, given the following code segment:

```
HotelReservation hotelReservation("Hotel Booking", 22, 9,
                                  "Mariott Hotel", 150.0, 2);

cout << "Hotel Reservation Details:" << endl;
cout << hotelReservation.to_string();
cout << "Total Cost: $" << hotelReservation.calculate_cost() << endl;
```

  Here's its expected output:

```
Hotel Reservation Details:
Hotel Booking on 09/22 for 2 nights at Mariott Hotel
Total Cost: $300
```