Topics: Composition and Inheritance

Classes to be considered during the given practice questions:

```
class personType
{
public:
void print() const;
//Function to output the first name and last name
//in the form firstName lastName.
void setName(string first, string last);
//Function to set firstName and lastName according
//to the parameters.
//Postcondition: firstName = first; lastName = last;
string getFirstName() const;
//Function to return the first name.
//Postcondition: The value of firstName is returned.
string getLastName() const;
//Function to return the last name.
//Postcondition: The value of lastName is returned.
personType(string first = "", string last = "");
//Constructor
//Sets firstName and lastName according to the parameters.
//The default values of the parameters are null strings.
//Postcondition: firstName = first; lastName = last;
private:
string firstName; //variable to store the first name
string lastName; //variable to store the last name
};

class dateType
{
public:
void setDate(int month, int day, int year);
//Function to set the date.
//The member variables dMonth, dDay, and dYear are set
//according to the parameters.
//Postcondition: dMonth = month; dDay = day;
// dYear = year
int getDay() const;
//Function to return the day.
//Postcondition: The value of dDay is returned.
int getMonth() const;
```

```
//Function to return the month.
//Postcondition: The value of dMonth is returned.
int getYear() const;
//Function to return the year.
//Postcondition: The value of dYear is returned.
void printDate() const;
//Function to output the date in the form mm-dd-yyyy.
dateType(int month = 1, int day = 1, int year = 1900);
//Constructor to set the date
//The member variables dMonth, dDay, and dYear are set
//according to the parameters.
//Postcondition: dMonth = month; dDay = day; dYear = year;
// If no values are specified, the default
// values are used to initialize the member
// variables.
private:
int dMonth; //variable to store the month
int dDay; //variable to store the day
int dYear; //variable to store the year
};
```

1. In this exercise, you will design various classes and write a program to computerize the billing system of a hospital.

a. Design the class **doctorType**, inherited from the class **personType** given below, with an additional data member to store a doctor's speciality. Add appropriate constructors and member functions to initialize, access, and manipulate the data members.

b. Design the class **billType** with data members to store a patient's ID and a patient's hospital charges, such as pharmacy charges for medicine, doctor's fee, and room charges. Add appropriate constructors and member functions to initialize and access and manipulate the data members.

c. Design the class **patientType**, inherited from the class **personType**, with additional data members to store a patient's ID, age, date of birth, attending physician's name, the date when the patient was admitted in the hospital, and the date when the patient was discharged from the hospital. (Use the class **dateType** to store the date of birth, admit date, discharge date, and the class doctorType to store the attending physician's name). Add appropriate constructors and member functions to initialize, access, and manipulate the data members.

d. Write a program to test your classes.

2.

a. Define the class **bankAccount** to store a bank customer's account number and balance. Suppose that account number is of type int, and balance is of type double. Your class should, at least, provide the following operations: set the account number, retrieve the account number, retrieve the balance, deposit and withdraw money, and print account information. Add appropriate constructors.

b. Every bank offers a checking account. Derive the class **checkingAccount** from the class bankAccount (designed in part (a)). This class inherits members to store the account number and the balance from the base class. A customer with a checking account typically receives interest, maintains a minimum balance, and pays service charges if the balance falls below the minimum balance. Add member variables to store this additional information. In addition to the operations inherited from the base class, this class should provide the following operations: set interest rate, retrieve interest rate, set minimum balance, retrieve minimum balance, set service charges, retrieve service charges, post interest, verify if the balance is less than the minimum balance, write a check, withdraw (override the method of the base class), and print account information. Add appropriate constructors.

c. Every bank offers a savings account. Derive the class **savingsAccount** from the class bankAccount (designed in part (a)). This class inherits members to store the account number and the balance from the base class. A customer with a savings account typically receives interest, makes deposits, and withdraws money. In addition to the operations inherited from the base class, this class should provide the following operations: set interest rate, retrieve interest rate, post interest, withdraw (override the method of the base class), and print account information. Add appropriate constructors.

d. Write a program to test your classes designed in parts (b) and (c).

3.

The class dateType was designed and implemented to keep track of a date, but it has very limited operations. Redefine the class **dateType** so that it can perform the following operations on a date, in addition to the operations already defined:

a. Set the month after validating the month.
b. Set the day after validating the day.
c. Set the year after validating the year.
d. Return the month.
e. Return the day.
f. Return the year.
g. Test whether the year is a leap year.
h. Return the number of days in the month. For example, if the date is 3-12-2013, the number of days to be returned is 31 because there are 31 days in March.

i. Return the number of days passed in the year. For example, if the date is 3-18-2013, the number of days passed in the year is 77. Note that the number of days returned also includes the current day.
j. Return the number of days remaining in the year. For example, if the date is 3-18-2013, the number of days remaining in the year is 288.
k. Calculate the new date by adding a fixed number of days to the date. For example, if the date is 3-18-2013 and the days to be added are 25, the new date is 4-12-2013.

4. The class dateType defined in Exercise 3 prints the date in numerical form. Some applications might require the date to be printed in another form, such as March 24, 2013. Derive the class **extDateType** so that the date can be printed in either form. Add a member variable to the class extDateType so that the month can also be stored in string form. Add a member function to output the month in the string format, followed by the year—for example, in the form March 2013. Write the definitions of the functions to implement the operations for the class extDateType.

5. Design and implement a class **dayType** that implements the day of the week in a program. The class dayType should store the day, such as Sun for Sunday. The program should be able to perform the following operations
on an object of type dayType:
a. Set the day.
b. Print the day.
c. Return the day.
d. Return the next day.
e. Return the previous day.
f. Calculate and return the day by adding certain days to the current day.
For example, if the current day is Monday and we add 4 days, the day to be returned is Friday. Similarly, if today is Tuesday and we add 13 days, the day to be returned is Monday.
g. Add the appropriate constructors.

6. Using the classes **extDateType** and **dayType** design the class **calendarType** so that, given the month and the year, we can print the calendar for that month.

To print a monthly calendar, you must know the first day of the month and the number of days in that month. Thus, you must store the first day of the month, which is of the form dayType, and the month and the year of the calendar. Clearly, the month and the year can be stored in an object of the form extDateType by setting the day component of the date to 1 and the month and year as specified by the user. Thus, the class calendarType has two member variables: an object of the type dayType and an object of the type extDateType.

Design the class calendarType so that the program can print a calendar for any month starting January 1, 1500. Note that the day for January 1 of the year 1500 is a Monday. To calculate the first day of a month, you can add the appropriate days to Monday of January 1, 1500.

For the class calendarType, include the following operations:
a. Determine the first day of the month for which the calendar will be printed. Call this operation firstDayOfMonth.
b. Set the month.
c. Set the year.
d. Return the month.
e. Return the year.
f. Print the calendar for the particular month.
g. Add the appropriate constructors to initialize the member variables.

b. Write a test program to print the calendar for either a particular month or a particular year.

7.