National University of Computer and Emerging Sciences



Assignment

For

Object Oriented Programming Lab

|  |  |
| --- | --- |
| Course Instructor(s) | Mr. Usman Ghous |
| Lab Instructor(s) | Mr. Usman Ghous |
| Semester | Spring 2021 |

**FAST School of Computing**

# Instructions:

1. Make a word document with the naming convention “SECTION\_ LAB#\_ROLLNO” and put all your source code and snapshots of its output in it. Make sure your word file is formatted properly.
2. **Plagiarism is strictly prohibited.**
3. Do not discuss solutions with one another.

**CLASSES AND DATA ABSTRACTION**

**QUESTION 1**: Write a program that converts a number entered in Roman numerals to  
decimal. Your program should consist of a class, say, romanType. An  
object of type romanType should do the following:  
a. Store the number as a Roman numeral.  
b. Convert and store the number into decimal form.  
c. Print the number as a Roman numeral or decimal number as requested.

M 1000  
D 500  
C 100  
L 50  
X 10  
V 5  
I 1  
d. Test your program using the following Roman numerals: MCXIV,  
CCCLIX, MDCLXVI.

**QUESTION 2:.** Design and implement a **class dayType** that implements the day of theweek in a program. The **class dayType** should store the day, such as **Sun**for Sunday. The program should be able to perform the following operationson 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 tobe 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. **QUESTION 3:**. Write the definitions of the functions to implement the operations for the  
class dayType as defined in Programming Exercise 2. Also, write a  
program to test various operations on this class.

**QUESTION 4**: This chapter defines the class clockType to implement time in a program. Add functions to this class so that a program that uses this class can set  
only the hours, minutes, or seconds and retrieve only the hours, minutes, or  
seconds. Also write a program to test your class.  
**QUESTION 5:** Example 11-9 defined a class personType to store the name of a person.  
The member functions that we included merely print the name and set the  
name of a person. Redefine the class personType so that, in addition to  
what the existing class does, you can:  
a. Set the first name only.  
b. Set the last name only.  
c. Store and set the middle name.  
d. Check whether a given first name is the same as the first name of this person.  
e. Check whether a given last name is the same as the last name of this person.  
Write the definitions of the member functions to implement the operations  
for this class. Also, write a program to test various operations on this class.  
**QUESTION 6:** a. Some of the characteristics of a book are the title, author(s), publisher,  
ISBN, price, and year of publication. Design a class bookType that  
defines the book as an ADT.  
i. Each object of the class bookType can hold the following  
information about a book: title, up to four authors, publisher,  
ISBN, price, and number of copies in stock. To keep track of  
the number of authors, add another member variable.

ii. Include the member functions to perform the various operations on  
objects of type bookType. For example, the usual operations that  
can be performed on the title are to show the title, set the title, and  
check whether a title is the same as the actual title of the book.  
Similarly, the typical operations that can be performed on the  
number of copies in stock are to show the number of copies in stock,  
set the number of copies in stock, update the number of copies in  
stock, and return the number of copies in stock. Add similar operations for the publisher, ISBN, book price, and authors. Add the  
appropriate constructors and a destructor (if one is needed).  
b. Write the definitions of the member functions of the class bookType.  
c. Write a program that uses the class bookType and tests various  
operations on the objects of the class bookType. Declare an array  
of 100 components of type bookType. Some of the operations that you  
should perform are to search for a book by its title, search by ISBN, and  
update the number of copies of a book.  
**QUESTION 7**: In this exercise, you will design a class memberType.  
a. Each object of memberType can hold the name of a person, member  
ID, number of books bought, and amount spent.  
b. Include the member functions to perform the various operations on the  
objects of memberType—for example, modify, set, and show a person’s  
name. Similarly, update, modify, and show the number of books bought  
and the amount spent.  
c. Add the appropriate constructors.  
d. Write the definitions of the member functions of memberType.  
e. Write a program to test various operations of your class memberType.  
**QUESTION 8**: Using the classes designed in Programming Exercises 6 and 7, write a  
program to simulate a bookstore. The bookstore has two types of customers:  
those who are members of the bookstore and those who buy books from the  
bookstore only occasionally. Each member has to pay a $10 yearly membership fee and receives a 5% discount on each book purchased.  
For each member, the bookstore keeps track of the number of books  
purchased and the total amount spent. For every eleventh book that a  
member buys, the bookstore takes the average of the total amount of the  
last 10 books purchased, applies this amount as a discount, and then resets the  
total amount spent to 0.  
Write a program that can process up to 1000 book titles and 500 members. Your  
program should contain a menu that gives the user different choices to effectively  
run the program; in other words, your program should be user driven.  
**QUESTION 9**: The method sellProduct of the Candy Machine programming example  
gives the user only two chances to enter enough money to buy the product.

Rewrite the definition of the method sellProduct so that it keeps prompting the user to enter more money as long as the user has not entered enough  
money to buy the product. Also, write a program to test your method.  
**QUESTION 10**: Write the definition of a class, swimmingPool, to implement the  
properties of a swimming pool. Your class should have the instance  
variables to store the length (in feet), width (in feet), depth (in feet), the  
rate (in gallons per minute) at which the water is filling the pool, and the  
rate (in gallons per minute) at which the water is draining from the pool.  
Add appropriate constructors to initialize the instance variables. Also add  
member functions to do the following: determine the amount of water  
needed to fill an empty or partially filled pool; determine the time needed  
to completely or partially fill or empty the pool; add or drain water for a  
specific amount of time.  
**QUESTION 11**: (Tic-Tac-Toe) Write a program that allows two players to play the tic-tac-toe  
game. Your program must contain the class ticTacToe to implement a  
ticTacToe object. Include a 3-by-3 two-dimensional array, as a private  
member variable, to create the board. If needed, include additional member  
variables. Some of the operations on a ticTacToe object are printing the  
current board, getting a move, checking if a move is valid, and determining the  
winner after each move. Add additional operations as needed.  
**QUESTION 12**: The equation of a line in standard form is ax + by ¼ c, wherein both a and  
b cannot be zero, and a, b, and c are real numbers. If b 6¼ 0, then –a/b is the  
slope of the line. If a ¼ 0, then it is a horizontal line, and if b ¼ 0, then it is  
a vertical line. The slope of a vertical line is undefined. Two lines are  
parallel if they have the same slope or both are vertical lines. Two lines are  
perpendicular if either one of the lines is horizontal and the other is vertical  
or the product of their slopes is –1. Design the class lineType to store a  
line. To store a line, you need to store the values of a (coefficient of x), b  
(coefficient of y), and c. Your class must contain the following operations.  
a. If a line is nonvertical, then determine its slope.  
b. Determine if two lines are equal. (Two lines a1x + b1y ¼ c1 and a2x +  
b2y ¼ c2 are equal if either a1 ¼ a2, b1 ¼ b2, and c1 ¼ c2 or a1 ¼ ka2,  
b1 ¼ kb2, and c1 ¼ kc2 for some real number k.)  
c. Determine if two lines are parallel.  
d. Determine if two lines are perpendicular.  
e. If two lines are not parallel, then find the point of intersection.  
Add appropriate constructors to initialize variables of lineType. Also write  
a program to test your class.

**QUESTION 13**: Typically, everyone saves money periodically for retirement, buying a  
house, or for some other purposes. If you are saving money for retirement,  
then the money you put in a retirement fund is tax sheltered, and the  
employer also makes some contribution into your retirement fund. In this  
exercise, for simplicity, we assume that the money is put into an account that  
pays a fixed interest rate, and money is deposited into the account at the end  
of the specified period. Suppose that a person deposits R dollars m times a  
year into an account that pays r % interest compounded m times a year for t  
years. Then the total accumulated at the end of t years is given by  
R ð Þ 1þr=m mt1  
h i r=m . For example, suppose that you deposit $500 at the end of  
each month into an account that pays 4.8% interest per year compounded  
monthly for 25 years. Then the total money accumulated into the account is  
500[(1 + 0.048/12)300 – 1]/(0.048/12) ¼ $289,022.42.  
On the other hand, suppose that you want to accumulate S dollars in t years  
and would like to know how much money, m times a year, you should  
deposit into an account that pays r% interest compounded m times a year.  
The periodic payment is given by the formula sðr=mÞ  
ð1þr=mÞmt1.  
Design a class that uses the above formulas to determine the total accumulated  
into an account and the periodic deposits to accumulate a specifc amount.  
Your class should have instance variables to store the periodic deposit, the  
value of m, the interest rate, and the number of years the money will be  
saved. Add appropriate constructors to initialize instance variable, functions to  
set the values of the instance variables, functions to retrieve the values of the  
instance variables, and functions to do the necessary calculations and output  
results.

**QUESTION 14:** In Chapter 11, the **class clockType** was designed to implement the timeof day in a program. Certain applications, in addition to hours, minutes, andseconds, might require you to store the time zone. Derive the **class  
extClockType** from the **class clockType** by adding a member variableto store the time zone. Add the necessary member functions and constructorsto make the class functional. Also, write the definitions of the memberfunctions and the constructors. Finally, write a test program to test your **class**. **QUESTION 15**: In this chapter, the class dateType was designed to implement the date  
in a program, but the member function setDate and the constructor d check whether the date is valid before storing the date in the member  
variables. Rewrite the definitions of the function setDate and the constructor so that the values for the month, day, and year are checked before  
storing the date into the member variables. Add a member function,  
isLeapYear, to check whether a year is a leap year. Moreover, write a  
test program to test your class.  
**QUESTION 16**: A point in the x-y plane is represented by its x-coordinate and y-coordinate.  
Design a class, pointType, that can store and process a point in the x-y  
plane. You should then perform operations on the point, such as setting the  
coordinates of the point, printing the coordinates of the point, returning the  
x-coordinate, and returning the y-coordinate. Also, write a program to test  
various operations on the point.  
**QUESTION 17**: Every circle has a center and a radius. Given the radius, we can determine  
the circle’s area and circumference. Given the center, we can determine its  
position in the x-y plane. The center of the circle is a point in the x-y plane.  
Design a class, circleType, that can store the radius and center of the  
circle. Because the center is a point in the x-y plane and you designed the  
class to capture the properties of a point in Programming Exercise 3, you  
must derive the class circleType from the class pointType. You  
should be able to perform the usual operations on the circle, such as setting  
the radius, printing the radius, calculating and printing the area and circumference, and carrying out the usual operations on the center. Also, write a  
program to test various operations on a circle.  
**QUESTION 18**: Every cylinder has a base and height, wherein the base is a circle. Design a  
class, cylinderType, that can capture the properties of a cylinder and  
perform the usual operations on the cylinder. Derive this class from the  
class circleType designed in Programming Exercise 4. Some of the  
operations that can be performed on a cylinder are as follows: calculate and  
print the volume, calculate and print the surface area, set the height, set the  
radius of the base, and set the center of the base. Also, write a program to test  
various operations on a cylinder.  
**QUESTION 19**: Using classes, design an online address book to keep track of the names,  
addresses, phone numbers, and dates of birth of family members, close  
friends, and certain business associates. Your program should be able to  
handle a maximum of 500 entries.  
a. Define a class, addressType, that can store a street address, city,  
state, and ZIP code. Use the appropriate functions to print and store the  
address. Also, use constructors to automatically initialize the member  
variables.  
b. Define a class extPersonType using the class personType (as  
defined in Example 11-9, Chapter 11), the class dateType (as designed  
in this chapter’s Programming Exercise 2), and the class addressType.  
Add a member variable to this class to classify the person as a family member, friend, or business associate. Also, add a member variable to store  
the phone number. Add (or override) the functions to print and store the  
appropriate information. Use constructors to automatically initialize the  
member variables.  
c. Define the class addressBookType using the previously defined  
classes. An object of the type addressBookType should be able to  
process a maximum of 500 entries.  
The program should perform the following operations:  
i. Load the data into the address book from a disk.  
ii. Sort the address book by last name.  
iii. Search for a person by last name.  
iv. Print the address, phone number, and date of birth (if it exists) of a  
given person.  
v. Print the names of the people whose birthdays are in a given month.  
vi. Print the names of all of the people between two last names.  
vii. Depending on the user’s request, print the names of all family  
members, friends, or business associates.  
**QUESTION 20**: In Programming Exercise 2, 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.  
b. Set the day.  
c. Set 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.

**QUESTION 21**: Write the definitions of the functions to implement the operations definedfor the **class dateType** in Programming Exercise 7. **QUESTION 22:** The **class dateType** defined in Programming Exercise 7 prints the date innumerical form. Some applications might require the date to be printed inanother form, such as March 24, 2013. Derive the **class extDateType** sothat the date can be printed in either form.Add a member variable to the **class extDateType** so that the monthcan also be stored in string form. Add a member function to output themonth in the string format, followed by the year—for example, in theform March 2013.Write the definitions of the functions to implement the operations for the **class extDateType**. **QUESTION 23**: Using the classes extDateType (Programming Exercise 9) and dayType  
(Chapter 11, Programming Exercise 2), 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.

**QUESTION 24:** a. Write the definitions of the member functions of the class  
calendarType (designed in Programming Exercise 10) to implement the operations of the class calendarType.

b. Write a test program to print the calendar for either a particular month  
or a particular year. For example, the calendar for September 2013 is:  
September 2013

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 |  |  |  |  |  |

**QUESTION 25**:

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, defined in Chapter 11, 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, defined in Chapter 11, 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.  
Write a program to test your classes.  
**QUESTION 26**:

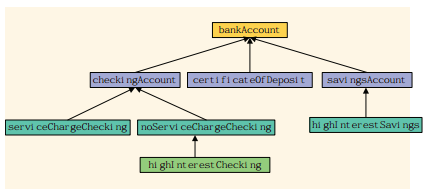
In the Programming Example Grade Report, in the definitions of the  
classes courseType and studentType, the accessor functions are not  
made constants; that is, they are not defined with the reserved word const  
at the end of their headings. Redefine these classes so that all of the accessor  
functions are constant functions. Accordingly, modify the definitions of the  
accessor functions and rerun the program.  
**QUESTION 27**:

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.

**POINTERS, CLASSES, VIRTUAL FUNCTIONS, ABSTRACT CLASSES, AND LISTS**

**QUESTION 28**: Redo Programming Exercise 5 of Chapter 9 using dynamic arrays. **QUESTION 29**: Redo Programming Exercise 6 of Chapter 9 using dynamic arrays.

**QUESTION 30**: Redo Programming Exercise 7 of Chapter 9 using dynamic arrays. You must  
ask the user for the number of candidates and then create the appropriate  
arrays to hold the data.  
**QUESTION 31**: Programming Exercise 11 in Chapter 9 explains how to add large integers using  
arrays. However, in that exercise, the program could add only integers of, at  
most, 20 digits. This chapter explains how to work with dynamic integers.  
Design a class named largeIntegers such that an object of this class can  
store an integer of any number of digits. Add operations to add, subtract,  
multiply, and compare integers stored in two objects. Also add constructors to  
properly initialize objects and functions to set, retrieve, and print the values  
of objects.  
**QUESTION 32**: Banks offer various types of accounts, such as savings, checking, certificate of deposits, and money market, to attract customers as well as meet  
with their specific needs. Two of the most commonly used accounts are  
savings and checking. Each of these accounts has various options. For  
example, you may have a savings account that requires no minimum  
balance but has a lower interest rate. Similarly, you may have a checking  
account that limits the number of checks you may write. Another type of  
account that is used to save money for the long term is certificate of  
deposit (CD).  
In this programming exercise, you use abstract classes and pure virtual  
functions to design classes to manipulate various types of accounts. For  
simplicity, assume that the bank offers three types of accounts: savings,  
checking, and certificate of deposit, as described next.  
Savings accounts: Suppose that the bank offers two types of savings  
accounts: one that has no minimum balance and a lower interest rate and  
another that requires a minimum balance and has a higher interest rate.  
Checking accounts: Suppose that the bank offers three types of checking  
accounts: one with a monthly service charge, limited check writing, no  
minimum balance, and no interest; another with no monthly service  
charge, a minimum balance requirement, unlimited check writing and  
lower interest; and a third with no monthly service charge, a higher  
minimum requirement, a higher interest rate, and unlimited check  
writing.  
Certificate of deposit (CD): In an account of this type, money is left for  
some time, and these accounts draw higher interest rates than savings or  
checking accounts. Suppose that you purchase a CD for six months. Then  
we say that the CD will mature in six months. Penalty for early withdrawal  
is stiff.  
Figure 13-25 shows the inheritance hierarchy of these bank accounts.



Note that the classes bankAccount and checkingAccount are abstract. That is,  
we cannot instantiate objects of these classes. The other classes in Figure 13-25  
are not abstract.  
bankAccount: Every bank account has an account number, the name of  
the owner, and a balance. Therefore, instance variables such as name,  
accountNumber, and balance should be declared in the abstract class  
bankAccount. Some operations common to all types of accounts are retrieve  
account owner’s name, account number, and account balance; make deposits;  
withdraw money; and create monthly statement. So include functions to  
implement these operations. Some of these functions will be pure virtual.  
checkingAccount: A checking account is a bank account. Therefore, it  
inherits all the properties of a bank account. Because one of the objectives of  
a checking account is to be able to write checks, include the pure virtual  
function writeCheck to write a check.  
serviceChargeChecking: A service charge checking account is a checking  
account. Therefore, it inherits all the properties of a checking account. For  
simplicity, assume that this type of account does not pay any interest, allows the  
account holder to write a limited number of checks each month, and does not  
require any minimum balance. Include appropriate named constants, instance  
variables, and functions in this class.  
noServiceChargeChecking: A checking account with no monthly service  
charge is a checking account. Therefore, it inherits all the properties of a  
checking account. Furthermore, this type of account pays interest, allows the  
account holder to write checks, and requires a minimum balance.

highInterestChecking: A checking account with high interest is a checking  
account with no monthly service charge. Therefore, it inherits all the properties  
of a no service charge checking account. Furthermore, this type of account pays  
higher interest and requires a higher minimum balance than the no service  
charge checking account.  
savingsAccount: A savings account is a bank account. Therefore, it inherits  
all the properties of a bank account. Furthermore, a savings account also pays  
interest.  
highInterestSavings: A high-interest savings account is a savings account.  
Therefore, it inherits all the properties of a savings account. It also requires a  
minimum balance.  
certificateOfDeposit: A certificate of deposit account is a bank account.  
Therefore, it inherits all the properties of a bank account. In addition, it has  
instance variables to store the number of CD maturity months, interest rate, and  
the current CD month.  
Write the definitions of the classes described in this programming exercise and a  
program to test your classes.  
**QUESTION 33**: The function retrieveAt of the class arrayListType is written as a  
void function. Rewrite this function so that it is written as a value returning  
function, returning the required item. If location of the item to be returned  
is out of range, use the assert function to terminate the program. Also, write  
a program to test your function. Use the class unorderedArrayListType  
to test your function.  
**QUESTION 34**: The function removeAt of the class arrayListType removes an element  
from the list by shifting the elements of the list. However, if the element to  
be removed is at the beginning of the list and the list is fairly large it could  
take a lot of computer time. Because the list elements are in no particular  
order, you could simply remove the element by swapping the last element of  
the list with the item to be removed and reducing the length of the list.  
Rewrite the definition of the function removeAt using this technique. Use  
the class unorderedArrayListType to test your function.  
**QUESTION 35**: The function remove of the class arrayListType removes only the first  
occurrence of an element. Add the function removeAll, as an abstract  
function, to the class arrayListType that would remove all occurrences  
of a given element. Also, write the definition of the function removeAll, in  
the class unorderedArrayListType, and a program to test this function.  
**QUESTION 36**: Add the function min, as an abstract function, to the class arrayListType  
to return the smallest element of the list. Also, write the definition of the  
function min, in the class unorderedArrayListType, and a program to  
test this function.

**QUESTION 37**: Add the function max, as an abstract function, to the class arrayListType  
to return the largest element of the list. Also, write the definition of the  
function max, in the class unorderedArrayListType, and a program to  
test this function.  
**QUESTION 38**: Write the definitions of the functions of the class orderedArrayListType,  
that are not given in this chapter. Also write a program to test various  
operations of this class.  
**QUESTION 39**: (Unordered Sets) As explained in this chapter, a set is a collection of  
distinct elements of the same type. Design the class unorderedSetType,  
derived from the class unorderedArrayListType, to manipulate sets.  
Note that you need to redefine only the functions insertAt, insertEnd,  
and replaceAt. If the item to be inserted is already in the list, the functions  
insertAt and insertEnd output an appropriate message. Similarly, if the  
item to be replaced is already in the list, the function replaceAt outputs an  
appropriate message. Also write a program to test your class.  
**QUESTION 40**: (Ordered Sets) Programming Exercise 12 asks you to define the class  
unorderedSetType to manipulate sets. The elements of an unorderedSetType  
object are distinct, but in no particular order.Design the class orderedSetType,  
derived from the class orderedArrayListType, to manipulate ordered sets.  
The elements of an orderedSetType object are distinct and in ascending order.  
Note that you need to redefine only the functions insert and replaceAt. If the  
item to be inserted is already in the list, the function insert outputs an appropriate  
message. Similarly, if the item to be replaced is already in the list, the function  
replaceAt outputs an appropriatemessage. Also write a program to test your class.

**OVERLOADING AND TEMPLATES**

**QUESTION 41**:

a. Write the definitions of the functions to overload the increment, decrement, arithmetic, and relational operators as members of the class  
rectangleType.  
b. Write a test program that tests various operations on the class  
rectangleType.  
**QUESTION 42**:

a. Write the definitions of the functions to overload the increment, decrement, arithmetic, and relational operators as nonmembers of the class  
rectangleType.  
b. Write a test program that tests various operations on the class  
rectangleType.  
**QUESTION 43**:

a. Extend the definition of the class clockType by overloading  
the post-increment operator function as a member of the class  
clockType.  
b. Write the definition of the function to overload the post-increment  
operator for the class clockType as defined in part a.  
**QUESTION 44**:

a. The increment and relational operators in the class clockType are  
overloaded as member functions. Rewrite the definition of the class  
clockType so that these operators are overloaded as nonmember functions. Also, overload the post-increment operator for the class  
clockType as a nonmember.  
b. Write the definitions of the member functions of the class clockType  
as designed in part a.  
c. Write a test program that tests various operations on the class as designed  
in parts a and b.  
**QUESTION 45**:

a. Extend the definition of the class complexType so that it performs  
the subtraction and division operations. Overload the operators subtraction and division for this class as member functions.  
If (a, b) and (c, d) are complex numbers:  
(a, b) - (c, d ) = (a - c, b - d ).  
If (c, d ) is nonzero:  
(a, b) / (c, d) = ((ac + bd ) / (c2 + d 2), (-ad + bc) / (c2 + d 2)).  
b. Write the definitions of the functions to overload the operators - and / as  
defined in part a.  
c. Write a test program that tests various operations on the class  
complexType. Format your answer with two decimal places.

**QUESTION 46**:

a. Rewrite the definition of the class complexType so that the arithmetic and relational operators are overloaded as nonmember functions.

b. Write the definitions of the member functions of the class complexType  
as designed in part a.  
c. Write a test program that tests various operations on the class  
complexType as designed in parts a and b. Format your answer with  
two decimal places.  
**QUESTION 47**:

a. Extend the definition of the class newString as follows:  
i. Overload the operators + and += to perform the string concatenation operations.  
ii. Add the function length to return the length of the string.  
b. Write the definition of the function to implement the operations defined  
in part a.  
c. Write a test program to test various operations on the newString objects.

**QUESTION 48**:

a. Rewrite the definition of the class newString as defined and  
extended in Programming Exercise 7 so that the relational operators  
are overloaded as nonmember functions.  
b. Write the definition of the class newString as designed in part a.  
c. Write a test program that tests various operations on the class  
newString.  
**QUESTION 49**:

Rational fractions are of the form a / b, in which a and b are integers and  
b 6¼ 0. In this exercise, by ‘‘fractions’’ we mean rational fractions. Suppose  
a / b and c / d are fractions. Arithmetic operations on fractions are defined  
by the following rules:  
a=b þ c=d ¼ ðad þ bcÞ=bd  
a=b c=d ¼ ðad bcÞ=bd  
a=b c=d ¼ ac=bd  
ða=bÞ=ðc=dÞ ¼ ad=bc; in which c=d 6¼ 0:  
Fractions are compared as follows: a / b op c / d if ad op bc, in which op is  
any of the relational operations. For example, a / b < c / d if ad < bc.  
Design a class—say, fractionType—that performs the arithmetic and  
relational operations on fractions. Overload the arithmetic and relational  
operators so that the appropriate symbols can be used to perform the  
operation. Also, overload the stream insertion and stream extraction operators for easy input and output.  
Write a C++ program that, using the class fractionType, performs  
operations on fractions.

Among other things, test the following: Suppose x, y, and z are objects  
of type fractionType. If the input is 2/3, the statement:  
cin >> x;  
should store 2/3 in x. The statement:  
cout << x + y << endl;  
should output the value of x + y in fraction form. The statement:  
z = x + y;  
should store the sum of x and y in z in fraction form. Your answer need  
not be in the lowest terms.  
**QUESTION 49**: Recall that in C++, there is no check on an array index out of bounds.  
However, during program execution, an array index out of bounds can  
cause serious problems. Also, in C++, the array index starts at 0.  
Design and implement the class myArray that solves the array index out  
of bounds problem and also allows the user to begin the array index starting  
at any integer, positive or negative. Every object of type myArray is an  
array of type int. During execution, when accessing an array component,  
if the index is out of bounds, the program must terminate with an appropriate error message. Consider the following statements:  
myArray<int> list(5); //Line 1  
myArray<int> myList(2, 13); //Line 2  
myArray<int> yourList(-5, 9); //Line 3  
The statement in Line 1 declares list to be an array of 5 components, the  
component type is int, and the components are: list[0], list[1], ...,  
list[4]; the statement in Line 2 declares myList to be an array of 11 components, the component type is int, and the components are: myList[2],  
myList[3], ..., myList[12]; the statement in Line 3 declares yourList  
to be an array of 14 components, the component type is int, and the  
components are: yourList[-5], yourList[-4], ..., yourList[0],  
..., yourList[8]. Write a program to test the class myArray.  
**QUESTION 50**: Programming Exercise 10 processes only int arrays. Redesign the class  
myArray using class templates so that the class can be used in any  
application that requires arrays to process data.  
**QUESTION 51**: Design a class to perform various matrix operations. A matrix is a set of numbers arranged in rows and columns. Therefore, every element of a matrix has a row position and a column position. If A is a matrix of five rows and six columns, we say that the matrix A is of the size 5 6 and  
sometimes denote it as A56. Clearly, a convenient place to store a matrix is  
in a two-dimensional array. Two matrices can be added and subtracted if  
they have the same size. Suppose A = [aij] and B = [bij] are two matrices of the size m n, in which aij denotes the element of A in the ith row and the  
jth column, and so on. The sum and difference of A and B are given by:  
A þ B ¼ ½aij þ bij  
A B ¼ ½aij bij  
The multiplication of A and B (A \* B) is defined only if the number  
of columns of A is the same as the number of rows of B. If A is of the size  
m n and B is of the size n t, then A \*B = [cik] is of the size m t and  
the element cik is given by the formula:  
cik ¼ ai1b1k þ ai2b2k þ þ ainbnk  
Design and implement a class matrixType that can store a matrix of any size.  
Overload the operators +, -, and \* to perform the addition, subtraction, and  
multiplication operations, respectively, and overload the operator << to output a  
matrix. Also, write a test program to test various operations on the matrices.  
**QUESTION 52**: a. In Programming Exercise 1 in Chapter 11, we defined a class  
romanType to implement Roman numbers in a program. In that  
exercise, we also implemented a function, romanToDecimal, to convert a Roman number into its equivalent decimal number.  
Modify the definition of the class romanType so that the member  
variables are declared as protected. Use the class newString, as  
designed in Programming Exercise 7, to manipulate strings. Furthermore, overload the stream insertion and stream extraction operators for  
easy input and output. The stream insertion operator outputs the  
Roman number in the Roman format.  
Also, include a member function, decimalToRoman, that converts the  
decimal number (the decimal number must be a positive integer) to an  
equivalent Roman number format. Write the definition of the member  
function decimalToRoman.  
For simplicity, we assume that only the letter I can appear in front of  
another letter and that it appears only in front of the letters V and X. For  
example, 4 is represented as IV, 9 is represented as IX, 39 is represented  
as XXXIX, and 49 is represented as XXXXIX. Also, 40 will be represented as XXXX, 190 will be represented as CLXXXX, and so on.  
b. Derive a class extRomanType from the class romanType to do  
the following: In the class extRomanType, overload the arithmetic  
operators +, -, \*, and / so that arithmetic operations can be performed on  
Roman numbers. Also, overload the pre- and post-increment and decrement operators as member functions of the class extRomanType.

To add (subtract, multiply, or divide) Roman numbers, add (subtract,  
multiply, or divide, respectively) their decimal representations and then  
convert the result to the Roman number format. For subtraction, if the  
first number is smaller than the second number, output a message saying  
that, ‘‘Because the first number is smaller than the second,  
the numbers cannot be subtracted’’. Similarly, for division, the  
numerator must be larger than the denominator. Use similar conventions  
for the increment and decrement operators.  
c. Write the definitions of the functions to overload the operators  
described in part b.  
d. Test your class extRomanType on the following program. (Include  
the appropriate header files.)  
int main()  
{  
extRomanType num1("XXXIV");  
extRomanType num2("XV");  
extRomanType num3;  
cout << "Num1 = " << num1 << endl;  
cout << "Num2 = " << num2 << endl;  
cout << "Num1 + Num2 = " << num1 + num2 << endl;  
cout << "Num1 \* Num2 = " << num1 \* num2 << endl;  
cout << "Enter two numbers in Roman format: ";  
cin >> num1 >> num2;  
cout << endl;  
cout << "Num1 = " << num1 << endl;  
cout << "Num2 = " << num2 << endl;  
num3 = num2 \* num1;  
cout << "Num3 = " << num3 << endl;  
cout << "--num3: " << --num3 << endl;  
cout << "++num3: " << ++num3 << endl;  
return 0;  
}  
**QUESTION 53**: Consider the class dateType given in Chapter 12. In this class, add the  
functions to overload the increment and decrement operators to increase  
the date by a day and decrease the date by a day, respectively; relational  
operators to compare two dates; and stream operators for easy input and  
output. (Assume that the date is input and output in the form MM-DD-YYYY.)  
Also write a program to test your class.  
**QUESTION 54**:. Programming Exercise 12, Chapter 11, describes how to design the class  
lineType to implement a line. Redo this programming exercise so that  
the class lineType:

1. Overloads the stream insertion operator, <<, for easy output.  
   b. Overloads the stream extraction operator, >>, for easy intput. (The line  
   ax + by ¼ c is input as (a, b, c).)  
   c. Overloads the assignment operator to copy a line into another line.  
   d. Overloads the unary operator +, as a member function, so that it returns  
   true if a line is vertical; false otherwise.  
   e. Overloads the unary operator -, as a member function, so that it returns  
   true if a line is horizontal; false otherwise.  
   f. Overloads the operator ==, as a member function, so that it returns  
   true if two lines are equal; false otherwise.  
   g. Overloads the operator ||, as a member function, so that it returns  
   true if two lines are parallel; false otherwise.  
   h. Overloads the operator &&, as a member function, so that it returns  
   true if two lines are perpendicular; false otherwise.  
   Write a program to test your class.  
   **QUESTION 55**: Consider the classes class cashRegister and dispenserType given in  
   the programming example in Chapter 11.  
   a. In the class class cashRegister, add the functions to overload the  
   binary operators + and – to add and subtract an amount in a cash  
   register; the relational operators to compare the amount in two cash  
   registers; and the stream insertion operator for easy output.  
   b. The class dispenserType, in the programming example in  
   Chapter 11, is designed to implement a dispenser to hold and release  
   products. In this class, add the functions to overload the increment  
   and decrement operators to increment and decrement the number of  
   items by one, respectively, and the stream insertion operator for easy  
   output.  
   Write a program to test the classes designed in parts a and b.  
   **QUESTION 56**: (Stock Market) Write a program to help a local stock trading company  
   automate its systems. The company invests only in the stock market. At the  
   end of each trading day, the company would like to generate and post the  
   listing of its stocks so that investors can see how their holdings performed  
   that day. We assume that the company invests in, say, 10 different stocks.  
   The desired output is to produce two listings, one sorted by stock symbol  
   and another sorted by percent gain from highest to lowest.  
   The input data is provided in a file in the following format:  
   symbol openingPrice closingPrice todayHigh todayLow  
   prevClose volume
2. For example, the sample data is:  
   MSMT 112.50 115.75 116.50 111.75 113.50 6723823  
   CBA 67.50 75.50 78.75 67.50 65.75 378233  
   ...  
   The first line indicates that the stock symbol is MSMT, today’s opening price was  
   112.50, the closing price was 115.75, today’s high price was 116.50, today’s low  
   price was 111.75, yesterday’s closing price was 113.50, and the number of shares  
   currently being held is 6723823.  
   The listing sorted by stock symbols must be of the following form:  
   \*\*\*\*\*\*\*\*\* First Investor's Heaven \*\*\*\*\*\*\*\*\*\*  
   \*\*\*\*\*\*\*\*\* Financial Report \*\*\*\*\*\*\*\*\*\*  
   Stock Today Previous Percent

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | Open | Close | High | Low | Close | Gain | Volume |
| ------ | ----- | ----- | ----- | ----- | -------- | ------- | ------ |
| ABC | 123.45 | 130.95 | 132.00 | 125.00 | 120.50 | 8.67% | 10000 |
| AOLK | 80.00 | 75.00 | 82.00 | 74.00 | 83.00 | -9.64% | 5000 |
| CSCO | 100.00 | 102.00 | 105.00 | 98.00 | 101.00 | 0.99% | 25000 |
| IBD | 68.00 | 71.00 | 72.00 | 67.00 | 75.00 | -5.33% | 15000 |
| MSET | 120.00 | 140.00 | 145.00 | 140.00 | 115.00 | 21.74% | 30920 |

1. Closing Assets: $9628300.00  
   -\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*  
   Develop this programming exercise in two steps. In the first step (part a), design and  
   implement a stock object. In the second step (part b), design and implement an  
   object to maintain a list of stocks.  
   a. (Stock Object) Design and implement the stock object. Call the class  
   that captures the various characteristics of a stock object stockType.  
   The main components of a stock are the stock symbol, stock price, and  
   number of shares. Moreover, we need to output the opening price,  
   closing price, high price, low price, previous price, and the percent  
   gain/loss for the day. These are also all the characteristics of a stock.  
   Therefore, the stock object should store all this information.  
   Perform the following operations on each stock object:  
   i. Set the stock information.  
   ii. Print the stock information.  
   iii. Show the different prices.  
   iv. Calculate and print the percent gain/loss.  
   v. Show the number of shares

a.1. The natural ordering of the stock list is by stock symbol.  
Overload the relational operators to compare two stock  
objects by their symbols.  
a.2. Overload the insertion operator, <<, for easy output.  
a.3. Because the data is stored in a file, overload the stream  
extraction operator, >>, for easy input.  
For example, suppose infile is an ifstream object and the input file  
was opened using the object infile. Further suppose that myStock is  
a stock object. Then, the statement:  
infile >> myStock;  
reads the data from the input file and stores it in the object myStock.  
(Note that this statement reads and stores the data in the relevant  
components of myStock.)  
b. Now that you have designed and implemented the class stockType  
to implement a stock object in a program, it is time to create a list of  
stock objects.  
Let us call the class to implement a list of stock objects stockListType.  
The class stockListType must be derived from the class listType, which you designed and implemented in the previous exercise.  
However, the class stockListType is a very specific class, designed  
to create a list of stock objects. Therefore, the class stockListType is no longer a template.  
Add and/or overwrite the operations of the class listType to  
implement the necessary operations on a stock list.  
The following statement derives the class stockListType from the  
class listType.  
class stockListType: public listType<stockType>  
{  
member list  
};  
The member variables to hold the list elements, the length of the list,  
and the max listSize were declared as protected in the class  
listType. Therefore, these members can be directly accessed in the  
class stockListType.  
Because the company also requires you to produce the list ordered by the  
percent gain/loss, you need to sort the stock list by this component.However,  
you are not to physically sort the list by the component percent gain/loss.  
Instead, you will provide a logical ordering with respect to this component.

To do so, add a member variable, an array, to hold the indices of the  
stock list ordered by the component percent gain/loss. Call this array  
sortIndicesGainLoss. When printing the list ordered by the component percent gain/loss, use the array sortIndicesGainLoss to  
print the list. The elements of the array sortIndicesGainLoss will  
tell which component of the stock list to print next.  
c. Write a program that uses these two classes to automate the company’s  
analysis of stock data.  
**QUESTION 57**: Write the definitions of the member functions of the classes  
arrayListType and unorderedArrayListType that are not given  
in this chapter. Also write a program to test your function.  
**QUESTION 58**: Write the definition of the class template orderedArrayList,  
derived from the class arrayListType, to implement an ordered list.  
As in Chapter 13, add the function insert in this class. Provide the  
definitions of the nonabstract functions. Also write a program to test your  
class.  
**QUESTION 59**: (Unordered Sets) Redo Programming Exercise 12 of Chapter 13 using templates.  
**QUESTION 60**: (Ordered Sets) Redo Programming Exercise 13 of Chapter 13 using  
templates.

**EXCEPTION HANDLING**

**QUESTION 61**: Write a program that prompts the user to enter a length in feet and inches  
and outputs the equivalent length in centimeters. If the user enters a  
negative number or a nondigit number, throw and handle an appropriate  
exception and prompt the user to enter another set of numbers.  
**QUESTION 62**: Redo Programming Exercise 7 of Chapter 8 so that your program handles  
exceptions such as division by zero and invalid input.  
**QUESTION 63**: Write a program that prompts the user to enter time in 12-hour notation.  
The program then outputs the time in 24-hour notation. Your program  
must contain three exception classes: invalidHr, invalidMin, and  
invalidSec. If the user enters an invalid value for hours, then the program  
should throw and catch an invalidHr object. Similar conventions for the  
invalid values of minutes and seconds.  
**QUESTION 64**: Write a program that prompts the user to enter a person’s date of birth in  
numeric form such as 8-27-1980. The program then outputs the date of  
birth in the form: August 27, 1980. Your program must contain at least two  
exception classes: invalidDay and invalidMonth. If the user enters  
an invalid value for day, then the program should throw and catch an  
invalidDay object. Similar conventions for the invalid values of month  
and year. (Note that your program must handle a leap year.)

**QUESTION 65**: Assume the definition of Exercise 4, which defines the struct movieType.  
Write a program that declares a variable of type movieType, prompts the  
user to input data about a movie, and outputs the movie data.  
**QUESTION 66**: Write a program that reads students’ names followed by their test scores.  
The program should output each student’s name followed by the test scores  
and the relevant grade. It should also find and print the highest test score  
and the name of the students having the highest test score.

Student data should be stored in a struct variable of type studentType,  
which has four components: studentFName and studentLName of type  
string, testScore of type int (testScore is between 0 and 100), and  
grade of type char. Suppose that the class has 20 students. Use an array of 20  
components of type studentType.  
Your program must contain at least the following functions:  
a. A function to read the students’ data into the array.  
b. A function to assign the relevant grade to each student.  
c. A function to find the highest test score.  
d. A function to print the names of the students having the highest test  
score.  
Your program must output each student’s name in this form: last name  
followed by a comma, followed by a space, followed by the first name; the  
name must be left justified. Moreover, other than declaring the variables and  
opening the input and output files, the function main should only be a  
collection of function calls.  
**QUESTION 67**: Define a struct, menuItemType, with two components: menuItem of  
type string and menuPrice of type double.

**QUESTION 68**: Write a program to help a local restaurant automate its breakfast billing  
system. The program should do the following:  
a. Show the customer the different breakfast items offered by the restaurant.  
b. Allow the customer to select more than one item from the menu.  
c. Calculate and print the bill.  
Assume that the restaurant offers the following breakfast items (the price  
of each item is shown to the right of the item):

|  |  |
| --- | --- |
| Plain Egg | $1.45 |
| Bacon and Egg | $2.45 |
| Muffin | $0.99 |
| French Toast | $1.99 |
| Fruit Basket | $2.49 |
| Cereal | $0.69 |
| Coffee | $0.50 |
| Tea | $0.75 |

1. Use an array, menuList, of the struct menuItemType, as defined in  
   Programming Exercise 2. Your program must contain at least the  
   following functions:  
   • Function getData: This function loads the data into the array  
   menuList.  
   • Function showMenu: This function shows the different items  
   offered by the restaurant and tells the user how to select the items.

• Function printCheck: This function calculates and prints the check.  
(Note that the billing amount should include a 5% tax.)  
A sample output is:  
Welcome to Johnny's Restaurant  
Bacon and Egg $2.45

|  |  |
| --- | --- |
| Muffin | $0.99 |
| Coffee | $0.50 |
| Tax | $0.20 |
| Amount Due | $4.14 |

Format your output with two decimal places. The name of each item in the  
output must be left justified. You may assume that the user selects only one  
item of a particular type.  
**QUESTION 69**: Redo Exercise 4 so that the customer can select multiple items of a  
particular type. A sample output in this case is:  
Welcome to Johnny's Restaurant  
1 Bacon and Egg $2.45

|  |  |
| --- | --- |
| 2 Muffin | $1.98 |
| 1 Coffee | $0.50 |
| Tax | $0.25 |
| Amount Due | $5.18 |

1. **QUESTION 70**: Write a program whose main function is merely a collection of variable  
   declarations and function calls. This program reads a text and outputs the  
   letters, together with their counts, as explained below in the function  
   printResult. (There can be no global variables! All information must  
   be passed in and out of the functions. Use a structure to store the information.) Your program must consist of at least the following functions:  
   • Function openFile: Opens the input and output files. You must pass  
   the file streams as parameters (by reference, of course). If the file does not  
   exist, the program should print an appropriate message and exit. The  
   program must ask the user for the names of the input and output files.  
   • Function count: Counts every occurrence of capital letters A-Z and  
   small letters a-z in the text file opened in the function openFile. This  
   information must go into an array of structures. The array must be  
   passed as a parameter, and the file identifier must also be passed as a  
   parameter.  
   • Function printResult: Prints the number of capital letters and small  
   letters, as well as the percentage of capital letters for every letter A-Z and  
   the percentage of small letters for every letter a-z. The percentages  
   should look like this: ‘‘25%’’. This information must come from an array  
   of structures, and this array must be passed as a parameter.

**QUESTION 71:** Write a program that declares a **struct** to store the data of a baseballplayer (player’s name, number of home runs, and number of hits). Declarean array of 10 components to store the data of 10 baseball players. Yourprogram must contain a function to input data and a function to outputdata. Add functions to search the array to find the index of a specificplayer, and update the data of a player. (You may assume that input data isstored in a file.) Before the program terminates, give the user the option tosave data in a file. Your program should be menu driven, giving the uservarious choices.

# BEST OF LUCK