GUJARAT TECHNOLOGICAL UNIVERSITY

Master in Computer Application (Integrated MCA)
Year II – (Semester-IV) (W.E.F. January2015)

Subject Name: C++ with Class Libraries

Subject Code: 4440601

---

1. **Create a program in "C++" for implementing polynomial operations according to the following instructions:**

(a) Create a Class "Polynomial" containing appropriate data member to store the coefficient and exponent of a single valued polynomial.
(b) Create a member function to read the values of the polynomial from the user.
(c) Create appropriate dynamic constructor that would allow allocating memory for the polynomial dynamically.
(d) Create a copy constructor that would enable copying one polynomial into another.
(e) Create an appropriate destructor for the class.
(f) Overload "+" to add two polynomials (e.g. $(3x^3 + 6x^2 + 9) + (4x^3 + 3x^2 - 5) = 7x^3 + 9x^2 + 4$) and the "*" operator to multiply a polynomial with a constant (e.g. $3 * (3x^3 + 6x^2 + 9) = 9x^3 + 18x^2 + 27$).
(g) Overload the "<<" operator to display the contents of the objects in the form of a polynomial.
(h) Implement the class using main function that would allow the users to create, display, add, and multiply polynomials through a menu driven program.
(i) Create a manipulator for the class that would set the precision for all the coefficients of the polynomial while displaying.
(j) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

2. **Create a program in "C++" for implementing a banking scenario according to the following instructions:**

(a) Create a class "account" containing acc_no(char[20]), balance (double) as data members, and set_account_no, deposit, and show_balance as member functions.
(b) Inherit the "account" class appropriately in two child classes named "savings", and "current". The "savings" class should contain interest_rate (float) as a data member, and set_interest_rate as member function. The "current" class should have overdraft_limit (double) as a data member and set_overdraft_limit as member function.
(c) Each class ("account", "savings", and "current") should contain appropriate parameterized constructors and destructor.
(d) Create a pure virtual function called withdraw in the "account" class and override the same in "savings", and "current" class such that the savings account does not have a balance less than 1000, while a current account does not have a negative value less that the overdraft limit.
(e) Use exception handling mechanism to ensure validity of deposit and withdraw operations.
(f) Overload the "<<" operator to display the balance of any type of account.
(g) Implement the classes using menu driven program in a main function that would allow the user to deposit, withdraw, and check balance in any type of account.
(h) Use runtime polymorphism to perform operations on any type of account.
(i) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.

3. Proper indentation.

3. Create a program in "C++" for implementing matrix operations according to the following instructions:

a. Create a class "matrix" containing data members including an integer pointer, and integer variables for storing the number of rows and columns.
b. Create appropriate member functions for the "matrix" class to read and display values of the matrix from the user.
c. Create appropriate constructors that would allow allocating memory for the matrix.
d. Create a copy constructor that would enable copying one object of the matrix class into another.

e. Create a destructor for the class that would de-allocate memory.
f. Overload the "+" and "*" operators to add and multiply two objects of the matrix class.
g. Overload the "<<" and ">>" operators for displaying and reading values of matrix from the user.

h. Implement the matrix class using main function that would allow the users to perform different operations like create matrix, display matrix, add two matrices, multiply two matrices through a menu driven program.
i. Use exception handling mechanism to ensure that two matrices can be added only if they are of the same dimension.
j. The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

4. Create a program in "C++" according to the following instructions:

(a) Create a class "person" containing name (char[30]), and contact_no (char[10]) as data members.

(b) Inherit the "person" class appropriately in two classes called "employee" and "customer". The employee class should contain designation (char[30]), and salary (double) as data members. The customer class should contain credit_limit (double), and outstanding_amount (double) as data members.
(c) Create appropriate member functions in the employee and customer class for reading and setting values for the data members.
(d) Each class ("person", "employee", and "customer") should contain appropriate parameterized constructors and destructor.
(e) Overload the "==" operator for the employee and customer class in order to check whether the values of two objects of the same class are same.
(f) Overload the "<<" operator the employee and customer class in order to display the data in a proper format.
(g) Use exception handling mechanism to ensure that the salary of employee, and credit_limit of customer do not store negative values.
(h) Implement the classes using main function that would allow the user to create objects of employee and customer class, and also set and display their values using a menu driven program.

(i) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

5. Create a program in "C++" according to the following instructions:

(a) Create two classes called "Time_12" and "Time_24" for storing time in 12 hour format (e.g. 6:20:30 PM) and 24 hour format (e.g. 18:20:30) respectively. Both the classes should contain hour (int), minute (int), and second (int) as data members. The "Time_12" class should also contain an additional data member to store a value indicating AM/PM.

(b) Also, create appropriate constructor(s), destructor, and member functions that would read and display values of the class from the user.

(c) Create each of the classes in separate namespace and include appropriate namespace while implementing the classes in the main function.

(d) Create proper type conversion mechanism for converting an object of Time_12 class to an object of Time_24 class and vice-versa.

(e) Overload the increment and decrement operators (only pre) in Time_24 classe that would respectively increment and decrement the time by 1 second.

(f) Use exception handling mechanism to ensure validity of values for objects of each class.

(g) Implement the class using main function that would allow the user to enter, display, and convert data for any Time through a menu driven program.

(h) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

6. Create a program in "C++" for implementing a queue according to the following instructions:

(a) Create a generic class "MyQueue" using template that can store elements of any data type. The class should contain a generic pointer, int size, int front, and int rear as data members.

(b) Create a default constructor and member functions EnQueue, and Dequeue for the class.

(c) Create appropriate parameterized constructors (including dynamic constructor) for the class.

(d) Create a destructor for the class to destroy a queue by de-allocating the assigned memory and resetting values of data members.

(e) Overload "==" operator to check if all the elements of two queues are same.

(f) Use exception handling mechanism to tackle overflow and underflow conditions in the queue.

(g) Implement the class using main function that would allow the users to create queues and perform operations to en-queue, de-Queue elements, and compare two queues for equality using a menu driven program.

(h) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

7. Create a program in "C++" for implementing a list of values according to the following instructions:

(a) Create a class "MyList" that can store n integer elements (the value of n is decided dynamically). The class should contain an int pointer, and int size as data members.

(b) Create a default constructor and member functions called "insert", and "delete" for the class that would insert and delete an element at a particular position.

(c) Create an appropriate parameterized constructor in order to allocate memory to the list dynamically, and a destructor to de-allocate the memory.

(d) Use exception handling mechanism to ensure validity of position of the elements to be inserted or deleted.

(e) Overload the "<<" operator to display the elements of the list.

(f) Use file handling mechanism to read and write the elements of the list in a text file.

(g) Implement the class using main function that would allow the user to create, insert, delete and display lists through a menu driven program.

(h) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

8. Create a program in "C++" for implementing a singly linked list according to the following instructions:

(a) Create appropriate class/classes for singly linked list along with the required data members.

(b) Create appropriate default and parameterized constructor(s) for the class/classes.

(c) Create appropriate destructor to de-allocate memory.

(d) Create member functions to insert, delete, and search elements in the linked list

(e) Use exception handling mechanism to validate insert operation.

(f) Overload the "+" operator that would merge two sorted linked lists.

(g) Overload the "<<" operator to display all the elements of the list.

(h) Implement the class/classes using main function that would allow the user to insert, delete, search, and display elements in the list and to merge two linked lists through a menu driven program.

(i) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

9. Create a program in "C++" for implementing a circular queue according to the following instructions:

(a) Create a class "Circular Queue" having front(int),rear(int) and Queue[size] as data members.
(b) Create a default constructor, and appropriate parameterized constructors (including dynamic constructor) for the class.
(c) Create member functions EnQueue, and Dequeue for the class.
(d) Create a destructor for the class to destroy a queue by de-allocating the assigned memory and resetting values of data members.
(e) Overload "==" operator to check if all the elements of two queues are same.
(f) Use exception handling mechanism to tackle overflow and underflow conditions in the queue.
(g) Overload the "<<" operator to display all the elements in the queue.
(h) Implement the class using main function that would allow the users to create queues and perform operations to en-queue, de-Queue, and display elements, and compare two queues for equality using a menu driven program.
(i) Create an option in the menu that would allow the user to read and write the data in a queue into a binary file.
(j) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

10. Create a program in "C++" for implementing a stack according to the following instructions:

(a) Create a Class "MyStack" containing ptr (int *), size (int), and top (int) as data members.
(b) Create member functions push, and pop for the class.
(c) Create appropriate dynamic constructors that would allow allocating memory and assigning values to the data members of the class dynamically.
(d) Create a copy constructor that would enable copying the data of one object of the class to another.
(e) Create an appropriate destructor for the class.
(f) Overload "--" operator for popping and push function for pushing elements in the stack.
(g) Use exception handling mechanism to deal with various exception conditions including overflow and underflow of the stack.
(h) Implement the class using main function that would allow the users to push, pop, display values in the stack through a menu driven program.
(i) Use file handling mechanism to store and retrieve objects of the class in a file.
(j) Modify the class using template to make it a generic stack (i.e. a stack that can store elements of any data type).
(k) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

11. Create a program in "C++" for implementing vector operations according to the following instructions:

(a) Create a class "MyVector" containing data members including an integer pointer to store the vector, and an integer variable for storing the number of elements of the vector.
(b) Create appropriate member functions for the class to read and display values of the vector from the user.
(c) Create appropriate constructors that would allow allocating memory for the vector.
(d) Create a copy constructor that would enable copying one object of the vector class into another.
(e) Create a destructor for the class that would de-allocate memory.
(f) Overload the "+" and "*" operators to add and multiply two objects of the class.
(g) Overload the "<<" and ">>" operators for displaying and reading values of vector from the user.
(h) Implement the class using main function that would allow the users to perform different operations like create vector, display vector, add two vectors, multiply two vectors through a menu driven program.
(i) Use exception handling mechanism to ensure that two vectors can be added or multiplied only if they are of the same dimension.
(j) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

12. Create a program in "C++" for performing operations on complex numbers according to the following instructions:

(a) Create a class "Complex" containing data members: int real, and int imaginary for storing the real and imaginary parts of a complex number.
(b) Create appropriate member functions for the class to read and display values of the complex number from the user.
(c) Create appropriate default and parameterized constructors for the class that would allow initializing values of the complex number.
(d) Create a copy constructor that would enable copying one object of the class into another.
(e) Create a destructor for the class that would de-allocate memory.
(f) Overload the "+" and "-" operators to add and multiply two objects of the class.
(g) Overload the "<<" and ">>" operators for displaying and reading values of complex numbers from the user.
(h) Implement the class using main function that would allow the users to perform different operations like create complex numbers, display complex numbers, add two complex numbers, multiply two complex numbers through a menu driven program.
(i) Use file handling mechanism to store and retrieve objects of the class in a file.
(j) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

13. Create a program in "C++" according to the following instructions:

(a) Create a class "person" containing name(char[30]), gender(char), and contact_no (char[10]) as data members.

(b) Inherit the "person" class appropriately in a class called "student". The student class should contain enrollment_no (char[10]), and date_of_admission (char[10]) as data members.

(c) Each class ("person", and "student") should contain appropriate default, and parameterized constructors.

(d) Create appropriate member functions in the student class for reading and setting values for the data members.

(e) Overload the "==" operator for the student class in order to check whether the date_of_admission of two students are same.

(f) Overload the "<<" operator in the student class in order to display the data in a proper format.

(g) Use exception handling mechanism to ensure that the validity of the value of gender in the person class and the value of date_of_admission in the student class.

(h) Implement the classes using main function that would allow the user to create an array of objects of student, and also set and display their values using a menu driven program.

(i) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

14. Create a program in "C++" for implementing a list of values according to the following instructions:

(a) Create a class "MyList" that can store n integer elements (the value of n is decided dynamically). The class should contain an int pointer, and int size as data members.

(b) Create a default constructor and an appropriate parameterized constructor in order to allocate memory to the list dynamically.

(c) Create a destructor to de-allocate memory.

(d) Create member functions called "mean", "median", and "mode" that would find the mean, median, and mode of values stored in the list.

(e) Generalize the class using template such that int or float elements can be stored in the list.

(f) Write a function to insert an element at the end of the list and the "– –" operator to delete an element from the end of the list.

(g) Overload the "<<" operator to display the elements of the list.

(h) Use file handling mechanism to read and write the elements of the list in a file.

(i) Implement the class using main function that would allow the user to create, insert, delete, find mean, find median, find mode, and display elements of list through a menu driven program.

(j) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

15. Create a program in "C++" according to the following instructions:

(a) Create a class "item" containing item_name(char[30]), price(double), unit(char[10]) and quantity(float) as data members.
(b) The class should contain appropriate default, and parameterized constructors.
(c) Create appropriate member functions in the item class for reading and setting values for the data members.
(d) Overload the ">" operator for the item class in order to check whether the price of one item is greater than the other.
(e) Overload the "<<" operator in the item class in order to display the data in a proper format.
(f) Create appropriate functions to sort an array of items in ascending and descending order based on name and price.
(g) Use exception handling mechanism to ensure the validity of the value of price and quantity (i.e. greater than zero) in the item class.
(h) Implement the class using main function that would allow the user to create an array of objects of item, and also set and display their values using a menu driven program. The menu should also contain options that would allow the user to sort the items in ascending and descending order based on item name or price.
(i) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

16. Create a program to manipulate strings in "C++" according to the following instructions:

(a) Create a class "MyString" containing len (int), str (char*) as data members.
(b) Create appropriate default, and parameterized constructors for the class.
(c) Create a copy constructor that would copy the contents of one object of the class to another.
(d) Create member functions for the class to find the length of the string, and to count the number of words in the string.
(e) Overload the "==" and "!=" operator for the class in order to check equality and inequality of two strings respectively.
(f) Overload the "<<" operator in the student class in order to display the contents of a string object.
(g) Create a manipulator to display the contents of an object of the class in upper case.
(h) Implement the class using main function that would allow the user to create objects of the string class, enter and view the values of the objects, compare two strings for equality and inequality, display the length of a string, and count the number of words in a string using a menu driven program.
(i) Use file handling mechanism to read and write the data in the object in to a file.
(j) The program should follow a good programming practice:
    1. Proper naming conventions for identifiers.
    2. Comments at appropriate places.
    3. Proper indentation.

17. Create a program in "C++" according to the following instructions:

(a) Create two classes called "Currency_INR" and "Currency_USD" for storing currency value of India and US respectively. Both the classes should contain value (double) as data member. The "Currency_INR" class should also contain a static data member conversion_rate(float).

(b) Create appropriate default and parameterized constructors for each class.

(c) Create each of the classes in separate namespace and include appropriate namespace while implementing the classes in the main function.

(d) Create proper type conversion mechanism for converting an object of Currency_INR class to an object of Currency_USD class and vice-versa.

(e) Overload the increment and decrement operators (both pre and post) for both the classes that would respectively increment and decrement the value by 1 unit.

(f) Create appropriate manipulators that would allow displaying the value of an object of Currency_INR in USD equivalent value.

(g) Use exception handling mechanism to ensure validity of values (non-negative) for objects of each class.

(h) Implement the class using main function that would allow the user to enter, display, and convert data for any type of currency through a menu driven program.

(i) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

18. Create a program in "C++" according to the following instructions:

(a) Create a class "engine" containing chassis_no(char[20]), and Fuel_Capacity(float) as data members.

(b) Use containership to create a class "car" containing an object of "engine" class as data member along with Registration_No(char[20]), Model(char[20]), and Price(double).

(c) Each class ("engine" and "car") should contain appropriate default and parameterized constructors.

(d) Create appropriate member functions for the car class in order to get and set the values of its data members.

(e) Create appropriate functions that would sort an array of Cars based on Price and Fuel_Capacity.

(f) Overload the "<<" and ">>" operators for the car class in order to display and read data from the user in a proper format.

(g) Use exception handling mechanism to ensure that the Price of Car, and Fuel_Capicity of Engine store only positive values.

(h) Implement the classes using main function that would allow the user to create an array of Cars, input their value, view the values, sort the array based on Price, and Fuel_Capicity using a menu driven program.

(i) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

19. Create a program in "C++" according to the following instructions:

(a) Create a class called "Distance" containing a data member called feet(int) and inch(int) for storing distance.

(b) Create appropriate default and parameterized constructors.

(c) Use vector class from STL to store n objects of class Distance and n values of built in data type double.

(d) Use Iterator to display contents of class vector

(e) Overload the + and – operators to add and subtract two distances.

(f) Create appropriate manipulators that would allow displaying the value of an object of Distance class in terms of Feet and Inch (e.g. 45 Inches = 3 Feet and 9 Inches).

(g) Use exception handling mechanism to ensure validity of values (non-negative) for objects of class.

(h) Implement the class using main function that would allow the user to enter, display data for distance through a menu driven program.

(i) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.

20. Create a program in "C++" related to coordinate geometry according to the following instructions:

(a) Create a class "Coordinate" containing X_Coordinate(int), and Y_Coordinate(int) as data members.

(b) Use containership to create a class "Triangle" containing an array of 3 object of "Coordinate" class as data member along with Area(double), and Perimeter(double).

(c) The Coordinate class should only have a parameterized constructor, while the Triangle class should have a default as well as a parameterized constructor.

(d) Create appropriate member functions for the Triangle class in order to get and set the values of its data members.

(e) Create appropriate functions that would calculate the area and perimeter of a Triangle. [Hint: Perimeter = Sum of all three sides of a triangle. Area = $\sqrt{s\,(s-a)(s-b)\,(s-c)}$, where s=$\frac{(a+b+c)}{2}$. a, b, and c are sides of the triangle.

(f) Overload the "<<" and ">>" operators for the Triangle class in order to display and read data from the user in a proper format.

(g) Use exception handling mechanism to ensure the validity of the values of coordinates of a Triangle. [Hint: The sum of any two side of a triangle is always greater than the third side.]

(h) Implement the Triangle class using main function that would allow the user to create a Triangle, input the values of its coordinates, view the values, view its area and perimeter using a menu driven program.

(i) Create an option in the menu that would allow the user to read and write an object of Triangle class into a binary file.

(j) The program should follow a good programming practice:

1. Proper naming conventions for identifiers.
2. Comments at appropriate places.
3. Proper indentation.