Object Oriented Programming

Unit - I control structures, structures, functions

1. Write a recursive C++ function to calculate m^n where m and n are numbers entered by the user.
2. Write an infinite loop that asks the user to enter a character and prints its ASCII value. The loop breaks only if the character entered is 'B'.
3. Write a C++ program that prints a letter grade for average marks entered on a five point scale. The Letter grades for average points 1 to 5 are A, B, C, D & E. In case of average marks 2.3 the letter grade will be B and for 2.7 it will be C. The letter grade for average mark less than 1 will be 'F'. Average marks above 5 should not be accepted. Use appropriate functions for input and output.
4. Write a program to find out whether the input number is Armstrong: Example: $153= 1^3 + 5^3 + 3^3$.
5. Write a program with C++ to create structure Classroom with appropriate variables and create array to store structure variables and display it.
6. Write a program with one command line argument to print reverse of the inputted number.

Unit - II Arrays and String

1. Write a program to allow the user to input a number of integers, and then stores them in an int array. Write a function called maxint() that goes through the array, element by element, looking for the largest one. The function should take as arguments the address of the array and the number of elements in it, and return the index number of the largest element. The program should call this function and then display the largest element and its index number.
2. Write a program to demonstrate swap(), erase(), replace(), append() and find() member function on String.
3. Implement a Stack using an int array.
4. Write a program for matrix multiplication.

Unit - III Operator Overloading, Object And Classes

1. Design a Class 'Friends' with the following entities:
    - Name of a person
    - the person's best friend
    - a counter indicating how many other people have this person as their best friend.

   Write a program that reads names and allocates a friend for each of them. Design a function 'set_counter' that sets the counter for each person. And finally prints the list of all names along with their friends name and their counters.
2. Write a C++ Program that contains a class 'Year'. The 'Year' class contains a function 'Leap' that accepts an year from the user and checks if it is a leap year. The class 'Year' should have appropriate functions for accepting the year from the user and printing the result.
3. Design a class having the following entities
    - Employee Name
    - Department
    - Manager's name
    - Salary

Write a program that accepts names and other details as specified above of at least 6-7 employees and prints a list of employees in the following fashion as and when required
    - All employees from a specific department
    - All employees whose manager is the same.
4. Write a C++ program to create class Complex with two member variables (real & imaginary). Overload operators + and - to add and subtract, respectively, two Complex numbers.
5. Create class Distance with feet(int) and inches(float) as member variables, add an overloaded '-' operator that subtracts two distances. It should allow statements like d1 = d2 - d3. Assume that the operator will never be used to subtract larger number from a smaller one (that is negative distances are not allowed).
6. Design an Employee class with at least three member variables and Write a static function show_total() to print total number of employees and show_emps() to print emp_id of each employee.
7. Write a program to create Time class with hours(int), minutes(int) and sec(int) as member variables. Add constructors and destructors to the class; also provide getdata() and putdata() member functions to accept and display the details.
8. Write a class Distance with feet(int), inches(float) as member variables and friend overloaded + operator to perform addition of two Distance objects.
9. Write a program to create class Time with hours and minutes as member variables and overload << operator.

Unit IV Inheritance

1. Implement a base class 'Person'. Derive classes 'Student' and 'Instructor' from 'Person'. A person has a name and a birthday. A student has a roll number, major subject and a minor subject and the instructor has a name, salary and a subject to teach. Write the class definitions, relevant constructors, destructors and member functions to input and print data.
2. Write a program for multi level inheritance where base class is Person derive class employee from Person class and also create scientist class derived from employee class add suitable two member variables in each class.
3. Write a program to create multi level inheritance with the base class Vehicle with type (String), manufacture_year(int) as member variables and TwoWheeler class with chasis_no(int) derived from Vehicle class and Scooty class with engine_capacity(String) derived from TwoWheeler class.
4. Write a program to create a base class Account having balance, acct_no, acct_holders_name, and address as fields. Create a class SavingsAccount derived from Account having field rate_of_interest. Create a class CurrentAccount derived from Account having field overdraft_limit. Write methods for withdrawing money, depositing money, and checking balance.

Unit - V Pointer

1. Write a function swap(), that swaps two numbers, using pointer arguments.
2. Write a program to create an array of 10 integer elements and sort array elements in ascending order using pointer.
3. Suppose you have a main() with three local arrays, all the same size and type (say float). The first two are already initialized to values. Write a function called addarrays() that accepts the addresses of the three arrays as arguments; adds the contents of the first two arrays together, element by

element; and places the results in the third array before returning. A fourth argument to this function can carry the size of the arrays. Use pointer notation throughout; the only place you need brackets is in defining the arrays.

Unit - VI Virtual Function and Polymorphism

1. Write a base class ‘Worker’ and derived classes ‘HourlyWorker’ and ‘SalariedWorker’. Every worker has a name and salary rate. Write a virtual function Calculate_Sal(int hours) that computes weekly pay for each worker. An hourly worker gets paid the hourly wage for the actual number of hours worked at the rate of Rs.250/- per hour, if hours is less than or equal to 40. If the hourly worker works more than 40 hours, the excess over 40 is paid double the rate. The salaried worker gets paid for 40 hours a week, no matter what the actual number of hours he has worked.
2. Write a program to create Base as base class and Derv1 and Derv2 as derived class from Base class with one member function named display() and show how virtual functions are accessed with pointers.

Unit - VII Streams and Files

1. Write a program to write primitive data to the text file and display it also on the console.
2. Write a program to perform input and output with integers on binary file edata.dat.
3. Create a class Employee and save some objects of the class in a binary file. Also read back the saved objects.

Unit - VIII Templates And Exceptions

1. Write a program to create exception on class Stack to handle exception for empty stack 10 when pop function is called on stack.
2. Write a program to create class Stack with push() and pop() member functions that works with integer elements. Throw Full exception if user tries to push an element to already full stack.
3. Write a program to create function template abs() which prints absolute of the int, long and float number passed as parameter to the abs() function.

Unit - IX The Standard Template Library

1. Write a program to demonstrate STL algorithm on List container.
2. Write a program that applies the sort() algorithm to an array of floating point values entered by the user, and displays the result.
3. Apply the sort() algorithm to an array of words entered by the user, and display the result. Use push_back() to insert the words, and the [] operator and size() to display them.