# List of Programs

| | | |
|---|---|---|
| 4 | W.A.P in C++ to show the working of **function overloading** by using a function named **calculateArea()** to calculate area of square, rectangle and triangle using different signatures as required. | 12-14 |
| 5 | Write a Program in C++ to demosntrate the concept of data abstraction using the concept of Class and Objects | 14-15 |
| 6 | Create a class called **Invoice** that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance<br>**Data Members** -<br>• partNumber (type String)<br>• partDescription (type String)<br>• quantity of the item being purchased (type int )<br>price_per_item (type double)<br>Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named **getInvoiceAmount()** that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to<br>0.0. Write a test application named **invoiceTest** that demonstrates class Invoice's capabilities. | 15-18 |
| 7 | Imagine a tollbooth with a class called **TollBooth**. The two data items are of type **unsigned int** and **double** to hold the total number of cars and total amount of money collected. A constructor initializes both of these data members to 0. A member function called **payingCar( )** increments the car total and adds 0.5 to the cash total. Another function called **nonPayCar( )** increments the car total but adds nothing to the cash total. Finally a member function called **display( )** shows the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car , and another to count a non paying car. Pushing the **ESC** key should cause the program to print out the total number of cars and total cash and then exit. | 19-21 |

| | | |
|---|---|---|
| 8 | Create a class called **Time** that has separate int member data for hours, minutes and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values. A member function should display it in 11:59:59 format. A member function named **add()** should add two objects of type time passed as arguments. A **main ( )** program should create two initialized values<br>together, leaving the result in the third time variable. Finally it should display the value of this third variable. | 21-22 |
| 9 | Create class **SavingsAccount.** Use a static variable **annualInterestRate** to store the annual interest rate for all account holders. Each object of the class contains a private instance variable **savingsBalance** indicating the amount the saver currently has on deposit. Provide method **calculateMonthlyInterest** to calculate the monthly interest by multiplying the **savingsBalance** by **annualInterestRate** divided by 12.This interest should be added to **savingsBalance**. Provide a static method **modifyInterestRate** that sets the **annualInterestRate** to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of<br>$2000.00 and $3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers. | 23-25 |
| 10 | Create a class **Complex** having two int type variable named **real** & **img** denoting real and imaginary part respectively of a complex number. Overload **+ , - , == operator** to add, to subtract and to compare two complex numbers being denoted by the two complex type objects. | 25-26 |
| 11 | Using the concept of operator overloading.Write a program to overload using with and without friend Function.<br>**a.** Unary –<br>**b.** Unary ++ preincrement, postincrement<br>c. Unary -- predecrement, postdecrement | 27-30 |
| 12 | Create a Base class that consists of private, protected and public data members and member functions.<br>Try using different access modifiers for inheriting Base class to the Derived class and create a table that summarizes the above three modes (when derived in public, protected and private modes) and shows the access specifier of the members of base class in the Derived class. | 30-32 |

| | | |
|---|---|---|
| 13 | Create a class called **Student** that contains the data members like **age, name, enroll_no, marks**. Create another class called Faculty that contains data members like **facultyName, facultyCode, salary, deptt, age, experience, gender**. Create the function **display()** in both the classes to display the respective information. The derived Class **Person** demonstrates multiple inheritance. The<br>program should be able to call both the base classes and displays their information. Remove the ambiguity (When we have exactly same variables or same methods in both the base classes, which one will be called?) by proper mechanism. | 33-35 |
| 14 | Create a base class called **shape**. Use this class to store two **double** type values that could be used to compute the area of figures. Derive two specific classes called **triangle** and **rectangle** from base shape. Add to the base class , a member function **get_data()** to initialize base class data members and another member function **display_area()** to compute and display the area of figures. Make **display_area()** as a **virtual function** and redefine this function in the derived class to suit their requirements. Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display the area.<br>Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangle and used as follows:<br>Area of rectangle = x * y<br>Area of triangle = ½ *x*y | 36-37 |

| | | |
|---|---|---|
| 15 | Create a base class called **CAL_AREA(Abstract).** Use this class to store float type values that could be used to compute the volume of figures. Derive two specific classes called **cone, hemisphere** and **cylinder** from the base **CAL_AREA**. Add to the base class, a member function **getdata ( )** to initialize base class data members and another member function display **volume( )** to compute and display the volume of figures. Make display **volume ( )** as a pure virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes. Remember values given as input will be and used as follows:<br>Volume of cone = $(1/3)\pi r^2 h$<br>Volume of hemisphere = $(2/3)\pi r^3$<br>Volume of cylinder = $\pi r^2 h$ | 38-40 |
| 16 | Implement a C++ program to demonstrate and understand Diamond problem. | 40-41 |
| 17 | Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. Write a program that can create a list (create a class list) of given type (int, float, char etc.) and perform insertion and deletion on list object. | 42 |
| 18 | C++ program to implement different methods of List, Vector and Map in STL (Standard Template Library) | 43-46 |