**BCSE102P STRUCTURED AND OBJECT-ORIENTED  
PROGRAMMING LAB  
CYCLE SHEET-2**

**PRACTICE PROBLEM SET-1-MODULE 5 & 6**  
**PRACTICE PROBLEM SET-2-MODULE 7 & 8**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING  
VELLORE INSTITUTE OF TECHNOLOGY,  
VELLORE**

**PRACTICE PROBLEM SET-1-MODULE 5 & 6**

|  |  |
| --- | --- |
| 1. **Write a code with Student as a base class, Mark as intermediate class and Result as a**   **derived class.**   * **The base class named Student with data members id, and name and getstu() and**   **putstu() are the methods to read and display the id and name.**   * **The intermediate class named Marks with data members m1, m2, m3 and getmarks()**   **and putmarks() are the methods to read and display the marks.**   * **The derived class named Result with total, average as a data members and show() as**   **the method to display the total and average of marks.** (**MODULE 6 - EASY)** |  |
| **Test case:1**  **Input=**  Enter ID:6789  Enter Name:Priya  Enter mark m1:67  Enter mark m2:78  Enter mark m3:89  Output=  ID:6789  Name: Priya  Mark1:67  Mark2:78  Mark3:89  The total is:234  The average is:78  **Test case:2**  Input=  Enter ID:4567  Enter Name:Kavitha  Enter mark m1:90  Enter mark m2:89  Enter mark m3:99  Output=  ID:4567  Name: Kavitha  Mark1:90  Mark2:89  Mark3:99  The total is:278  The average is:92  **Test case:3**  **Input=**  Enter ID:345  Enter Name:Raj  Enter mark m1:89  Enter mark m2:56  Enter mark m3:34  Output=  ID:345  Name: Raj  Mark1:89  Mark2:56  Mark3:34  The total is:179  The average is:59  **Invalid test:**  Input=  Name: Ravi  Roll No: 89  Grade: 11  Output=  Error |  |
| 2. **Create a parent class named ‘Courier’ with the following:**   1. **Data members CourierID, Name\_of\_Courier.** 2. **Method named PrintBill() to accept the Base\_fare of type double as parameter and display the CourierID, Name\_of\_Courier , and    Shipping\_Cost, where Shipping\_Cost = Base\_fare + 30.**  * **Create a subclass names ‘Internatial\_services’ which inherits from the Courier class. The class include the following:**  1. **Data members Destination, Weight.** 2. **Method named FinalBill()to print the CourierID, Name\_of\_Courier, Destination, Weight and Total\_ ShippingCost, where Total\_ShippingCost = Base\_fare \* Weight. Print the message “More Sale” when Total\_ShippingCost is more than 100, otherwise print the message “Less Sale”.** (**MODULE 6 - EASY)** |  |
| **Test case:1**  Input=  Enter the Courier ID:1234  Enter the Name of the Courier: Professional  Enter the Base Fare:34  Enter the Weight in KG:100  Output=  CourierID: 1234  Name\_of\_Courier: Professional  The Shipping cost is:64  Total Shipping Cost=3400  More Sale  **TestCase2:**  Input=  Enter the Courier ID:5678  Enter the Name of the Courier: DTCCourier  Enter the Base Fare:3  Enter the Weight in KG:12  Output=  CourierID: 5678  Name\_of\_Courier: DTCCourier  The Shipping cost is:33  Total Shipping Cost=36  Less Sale  **TestCase3:**  Input=  Enter the Courier ID:8976  Enter the Name of the Courier: DHLExpress  Enter the Base Fare:78  Enter the Weight in KG:95  Output=  CourierID: 8976  Name\_of\_Courier: DHLExpress  The Shipping cost is:108  Total Shipping Cost=7410  More Sale  **Invalid test case:**  Input=  Enter the Courier ID:8976  Enter the Name of the Courier: DTCCourier  Enter the Base Fare:78  Enter the Weight in KG:600KG  Output=  Error |  |
| 3. **Create two classes DM and DB. DM stores the distance in meters and centimetres and DB Stores the distance in feet and inches. Write a program to read the values for the class**  **objects and Add one object DM with another object DB. Note: use a friend function to**  **carry out addition operation. The resultant object is stored in DM and display it.**  **1 feet = 30 centimetres**  **1 inch =2.54 centimetres**  **1 meter = 100 centimetres** (**MODULE 6 - EASY)** |  |
| **Test case:1**  **Input=**  Enter the value in meter and centimeter:  Enter meter value : 12  Enter centimeter value: 34  Enter the value in feet and inches:  Enter feet value : 12  Enter inches value : 4  Output=  The summed value in meter is: 15.648  **Test case:2**  Input=  Enter the value in meter and centimeter:  Enter meter value : 15  Enter centimeter value: 54  Enter the value in feet and inches:  Enter feet value :34  Enter inches value :7  Output=  The summed value in meter is: 25.336  **Test case:3**  Input=  Enter the value in meter and centimeter:  Enter meter value : 45  Enter centimeter value: 78  Enter the value in feet and inches:  Enter feet value :23  Enter inches value :9  Output=  The summed value in meter is: 51.992 |  |
| 4. Write a C++ program to read and print employee salary details using multilevel  inheritance.  a. Create a class named employee to get and print the employee details like employee number, name and designation.  b. Create a derived class named salary which derives the class employee in private mode, to get the employee complete information including employee number, name, designation, basic pay, hra, da, pf and to display the complete employee information including the net pay.  c. Create a derived class bank\_details which derives the class salary in private mode to get the complete employee details including his bank name, account number and to display the complete employee details like empno, name, designation, bp, hra, da, pf, net pay, account number.  d. Create appropriate main method for the same.(**MODULE 6 - MODERATE)** |  |
| **Test case:1**  **Input=**  enter employee no 1234  enter employee name abc  enter designation Sr.Engg  enter hra, da, pf  12  4  34  Enter bank name Indian  Enter account number 134512  Output=  Emp number:1234  Emp Name:abc  Designation:Sr.Engg  hra12  da4  pf34  Bank Name: Indian  Account Number: 134512  **Test case:2**  Input=  enter employee no 156789  enter employee name riya  enter designation manager  enter hra, da, pf  67  89  78  Enter bank name SBI  Enter account number 390934567  Output=  Emp number:156789  Emp Name:riya  Designation:manager  Hra67  Da89  Pf78  Bank Name: SBI  Account Number: 390934567  **Test case:3**  Input=  enter employee no 6789  enter employee name priya  enter designation teamleader  enter hra, da, pf  45  56  67  Enter bank name Canarabank  Enter account number 7896543  Emp number:156789  Emp Name:riya  Designation: manager  Hra67  Da89  Pf78  Bank Name: SBI  Account Number: 390934567 |  |
| 5. Use hierarchical inheritance to implement a student database that will be composed of classes Student, UnderGraduate, and PostGraduate.   * The base class ‘Student’ will have a data member Name, Id and Age. Getstu() and   Putstu() methods will ask the user to enter the details of the student and display the  same.   * In derived class, UnderGraduate will have the data member UGCourses to specify the   number of courses offered. GetUGCourses() and PetUGCourses() methods will ask  the user to enter the number of courses and display the same.   * In derived class, PostGraduate will have the data member PGCourses to specify the   number of courses offered. GetPGCourses() and PetPGCourses() methods will ask the  user to enter the number of courses and display the same. (**MODULE 6 - MODERATE)** |  |
| **Test case:1**  Input=  Enter the Name:Priya  Enter the ID:678  Enter the Age:21  Enter the Number of UG courses offered: 3  Enter the Number of PG courses offered: 5  Output:  Name: Priya  ID: 678  Age: 21  Number of UG courses offered: 3  Number of PG courses offered: 5  **Test case:2**  Input=  Enter the Name:Kavitha  Enter the ID:789  Enter the Age:24  Enter the Number of UG courses offered: 5  Enter the Number of PG courses offered: 2  Output:  Name: Kavitha  ID: 789  Age: 24  Number of UG courses offered: 5  Number of PG courses offered: 2  **Test case:3**  Input=  Enter the Name:Raghu  Enter the ID:345  Enter the Age:22  Enter the Number of UG courses offered: 2  Enter the Number of PG courses offered: 2  Output:  Name: Raghu  ID: 345  Age: 22  Number of UG courses offered: 2  Number of PG courses offered: 2  **Invalid test case**  Input=  Enter the Name:Priya  Enter the ID:678  Enter the Age:21  Enter the Number of UG courses offered: 10  Enter the Number of PG courses offered: 10  Output=  Error |  |
| **6.** Suppose you are working on a project to create a simple banking system and you need to  create a class hierarchy for different types of bank accounts. You want to create a base  class called BankAccount that contains basic account information and methods that are  common to all types of bank accounts. You also want to create a derived class called  SavingsAccount that inherits from BankAccount and contains additional methods specific  to savings accounts. Create a C++ code snippet for the above scenario that demonstrates  inheritance. (**MODULE 6 - MODERATE)** |  |
| **Test Case:**  Input=  Account number: 12345  Account holder name: John Doe  Balance: 10500  Interest rate: 5%  Minimum balance: 5000  Account number: 12345  Output=  Account holder name: John Doe  Balance: 8500  Interest rate: 5%  Minimum balance: 5000  Account number: 12345  Account holder name: John Doe  Balance: 13500  Interest rate: 5%  Minimum balance: 5000  **Invalid test case**  Input=  Account number: 12345  Account holder name: Priya  Balance: 10500  Interest rate: 5%  Minimum balance: 3000  Output=  Error |  |
| **7.** Consider creating a program to simulate various vehicle types. In addition to making a  basic class named Vehicle that has universally applicable properties and methods, you  also want to make distinct classes for various kinds of vehicles, such as cars and  airplanes. The distinct traits and procedures that distinguish each vehicle class from other  sorts of vehicles are their own. Nonetheless, several vehicle types might have things in  common and work in similar ways. A plane, for instance, also has an altitude, while both  a car and a plane have a maximum speed and a present speed. you can create a Vehicle  class that contains general properties and methods, and then create specific vehicle  classes that inherit from Vehicle as well as from each other. Create a C++ code snippet for  the above scenario that demonstrates inheritance. (**MODULE 6 - HARD)** |  |
| **Test case:1**  Input=  Enter Speed:78  Enter altitude:89  Output=  Flying!  Flying car speed: 78  Flying car altitude: 89  **Test case:2**  Input=  Enter Speed:90  Enter altitude:1000  Output=  Flying!  Flying car speed: 90  Flying car altitude: 1000  **Invalid test case:**  Input=  Enter Speed:130  Enter altitude:1000  Output=  Error |  |
| 8. In a school, there is a hierarchy of classes for the students with a base class of student and they had splitted the students into Junior Student and Senior Student. Senior Student has an additional class inheritance from a grade class, this grade class reflects the grades a student they earned in their exams. Create a C++ code snippet for the above scenario that demonstrates inheritance. (**MODULE 6 - HARD)** |  |
| **Test case:**  **Input=**  Name: John  Roll No: 101  Grade: 9  Output=  Name: Jane  Roll No: 201  Marks: 85  **Test case:**  **Input=**  Name: priya  Roll No: 107  Grade: 8  Output=  Name: riya  Roll No:601  Marks: 95  **Invalid test:**  Input=  Name:ravi  Roll No: 89  Grade: 11  Ouput=  Error |  |
| **9.** Write a Program to perform the following scenario. Define a class student with rollno as  member and getdata() and putdata() as member functions. .Define another class test that inherit the class student, this class test has data members as marks in the subject 1 and subject 2 with member functions getmark() and putmark(). Define a class Grade with data member grade and a member function to display the grade. Define another class result that inherit the classes test and Grade with data member total and member function to compute the total marks along with the grade.  **Note make use of the access specifier Private for Grade class** (**MODULE 6 - HARD)** |  |
| **Test case:1**  Input=  Enter rollno:1234  enter marks1 and marks2:  67  78  Enter Grade A  Output=  RollNO: 45  Marks1: 67  Marks2: 78  GradeA  Total:145  **Test case:2**  Input=  Enter rollno:4567  enter marks1 and marks2:  56  78  Enter Grade B  Output=  RollNO: 4567  Marks1: 56  Marks2: 78  GradeA  Total:134  **Invalid test case**  **Input=**  Enter rollno:789  enter marks1 and marks2:  101  102  Output=  Error |  |

|  |
| --- |
| **Module 5** |
| * + 1. Create a class called "Rectangle" that has attributes "length" and "width". Write a method that calculates the area of the rectangle.   (MODULE 5 - EASY) |
| Sample Test Cases:  Test Case 1- Rectangle r1(5, 10);  r1.calculateArea(); // Expected output: 50  Case2 - Rectangle r2(3, 8);  r2.calculateArea(); // Expected output:24  Case3-Rectangle r3(6 2)  r3.calculateArea();//Expected output:12  **Invalid test Case3**  Rectangle r3(0, ∞)  r3.calculateArea();//  Expected output: Invalid |
| * + 1. Write a program that takes in a number and outputs whether it is even or odd.Using C++. (MODULE 5 - EASY) |
| Test case1:  Sample input: 7  Sample output: Odd  Test Case2:  Sample input:10  Sample output:Even  Test case3:  Sample Input: 19  Sample output: odd  **Invalid Test case4:**  Sample input: -0.9876543  Sample output: Invalid |
| * + 1. Write a program that takes in two numbers and outputs the sum of their squares. Using C++ (MODULE 5 - EASY) |
| Sample input: 3, 4  Sample output: 25  Sample input: 1, 4  Sample output:17  Sample input : 5, 6  Sample output: 61  **Invalid Test case**  Sample input:a,b  Sample output: Invalid |
| * + 1. A car rental company wants to keep track of its fleet of cars. Each car has a make, model, year, and rental price. The company wants to be able to calculate the total rental price of all its cars and also find the car with the highest rental price. Design a program using OOPs and the "this" pointer to implement this scenario. Create a new car object and verify that all the attributes are set correctly using the "this" pointer. (MODULE 5 - MEDIUM) |
| Test case1  Input:  Model  Output:  Swift  Test Case2:  Input:  Year  Output:  2022  Test case3:  Input  Rental Price:  Output:  15000 |
| * + 1. Create a class called "Rectangle" that has private attributes "length" and "width" and public methods "getArea" and "getPerimeter" that return the area and perimeter of the rectangle. Write a program that creates an object of the class and tests the methods. (MODULE 5 - MEDIUM) |
| Sample input: length = 4, width = 5  Sample output: 20  Sample input:length=5, width=10  Sample output=50  **Invalid test case**  Sample input: length=-5, width=-10  Sample output=Invalid. |
| * + 1. Create a class called "Car" with attributes "make", "model", and "year". Create an object of the class and print out its attributes.   (MODULE 5 - MEDIUM) |
| Test case 1:  Car("Toyota", "Camry", 2022)  Output: Make: Toyota, Model: Camry, Year: 2022  Test case 2: Car("Honda", “Verna”, 2021)  Output: Make: Honda,Model:Verna, Year:2021  Test case3  Car("Suzuki", "swift", 2020)  Output: Make: Suzuki, Model: Swift, Year: 2020  **Invalid Test case3**  Car("Suzuki", "Swift", -2020)  Output: invalid |
| * + 1. Write a program that uses an inline function to calculate the area of a circle. The function should take in the radius as a parameter and return the area. Test the function with the following values: (MODULE 5 - HARD) |
| Case1- Radius = 5, Expected Output = 78.54  Case2- Radius = 10, Expected Output = 314.16  Case3- Radius = 2.5 , Expected output=19.64  **Invalid test case**  Case4-Radius= -9.5, Expected output: Invalid. |
| * + 1. A library has multiple books and each book has a title, author, and ISBN number. The library wants to keep track of the books that are currently available and the books that have been borrowed by the members. Design a program using OOPs and class and object to implement this scenario. (MODULE 5 - HARD) |
| Test Case 1:  Input:  Book Title:  Output:  Output:  "The Alchemist"  Test Case2:  Input:  Author Name:  Output:  "Paulo Coelho"  Input:  ISBN Number:  Output:  "978-006231 |
| Write a class named "Person" with a constructor that takes in a name and age as arguments. The constructor should initialize the object's name and age properties. Write a test case to verify that the constructor works correctly. . (**MODULE 5 - HARD)** |
| Test case1  Person p("John", 25);  assert(p.getName() == "John");  assert(p.getAge() == 25);  Test Case2  Person p(“sam”,35);  Assert(p.getName()==“sam”);  assert(p.getAge()==35);  Test Case3:  person p(“Ram”,45);  Assert(p.getName()==“Ram”);  assert(p.getAge()==45);  **Invalid Test Case3:**  Input: person p(“$”,-45);  invalid |
| **Module 7** |
| 1. Create a class string that reads the string and its length.  Write a program to overload the operator < and > to compare two strings.  if s1 i< s2 then print "String s1 is smaller than String s2", if s2< s1 then print "String s2 is smaller than String s1" otherwise print "Both the Strings s1 and s2 are Equal".  **(Module 7, Easy)** |
| Case1:  Input:  Arun  Bajaj  Output:  String s1 is smaller than String s2  Case2:  Input:  Hariharan  Antony  Output:  String s2 is smaller than String s1  Case3:  Input:  Kavin  Kavin  Output:  Both the Strings s1 and s2 are Equal |
| 1. Write a program to create a Class Matrix with data members row and columns along with the suitable member functions to read and display the matrix.  Define a function, + and - that overloads its operation by performing the operations matrix addition and subtraction.  Implement using the objects.  **[Module 7, Moderate]**   Case1:  Input:  2 3  2 2 3  4 5 6  2 3  4 5 6  1 2 3  +  Output:  6 7 9  5 7 9  Case2:  2 2  7 2  5 6  2 2  6 1  1 2  -  Output:  1 1  4 4  **Invalid Test Case3:**  2 2 3  7 2 4  5 6  2 2  -  Output:  Invalid Matrix Size |
| 1. Write a Program to read an odd number N. and overload the Prefix ++ increment operator and Prefix -- decrement operator.  **[Module 7, Easy]**   Note: Let us assume that N=11 then ++N should print N=13 and --N should print N=9  Case1:  Input:  15  ++  Output:  N=17  Case2:  Input:  25  --  Output:  N=23  **Invalid test Case3**:  Input:  0  --  Output=  Negative value |
| 1. An education centre runs several batches for various courses in a day. Due to the limited number of resources, they want to finalize the minimum number of class rooms needed to run the classes so that no batch of students wait.   Given the start time and end time of all batches, design a system that finds the minimum number of class rooms required for the education centre. Your system should overload the appropriate operators (wherever if possible).  Problem Input: List of start time and end time  Problem Output: Minimum number of class rooms required  **[Module 7, Hard]** |
| Case1:  Input: lectures [][] = {{0, 5}, {1, 2}, {1, 10}}  Output: 3  Case2:  Input: lectures [][] = {{0, 5}, {1, 2}, {6, 10}}  Output: 2 |
| 1. Three are three types of employees in Indian railways. They are regular, daily wages and consolidated employees. Gross Pay for the employees is calculated as follows using function overloading:  * Regular employees: Gross salary = Basic + HRA+ DA * Daily wages – Wages per hour \* Number of hours * Consolidated – fixed amount   Input: Components for calculating gross pay  Output: Gross pay **[Module 7, Hard]** |
| Case 1:  Input:  Basic=20000  HRA=9287  DA=2850  Wages per hour = 100  Number of hours = 7  Consolidated Pay=20000  Output:  Regular employees = 32137  Daily wages = 700  Consolidated = 20000  Case 2:  Input:  Basic=22000  HRA=10287  DA=2750  Wages per hour = 120  Number of hours = 7  Consolidated Pay=20000  Output:  Regular employees = 35037  Daily wages = 840  Consolidated = 20000  **Invalid Case 3:**  Input:  Basic=2000  HRA=18287  DA=2750  Wages per hour = 120  Number of hours = 7  Consolidated Pay=20000  Output:  Invalid Pay Scale |
| 1. In your main() , declare two Point objects that are initialized by the default constructor. Prompt user inputs as shown in the test case and save the user inputs with your overloaded >> operator. And display the x and y coordinates of the user inputs with your overloaded << operator as shown in the test case. **[Module 7, Moderate]** |
| Case1:  Input:  Enter x and y coordinates for first point, separated by a space: -3.4 8  Enter x and y coordinates for second point, separated by a space: 22 20.7  Output:  First point is at (-3.4, 8)  Second point is at (22, 20.7)  **InvalidtestCase2:**  Input:  Enter x and y coordinates for first point, separated by a space: 0 0  Enter x and y coordinates for second point, separated by a space: 0 0  Output:  Invalid Co-ordinates |
| 1. Create an abstract class MathSymbol may provide a pure virtual function doOperation(), and create two more classes Plus and Minus implement doOperation() to provide concrete implementations of addition in Plus class and subtraction in Minus class.  **[Module 7, Moderate]** |
| Case 1:  Input:  Enter the value of x and y:  31  24  Output:  Addition: 55  Subtraction: 7  Case 2:  Input:  Enter the value of x and y:  55  44  Output:  Addition: 99  Subtraction: 11  **Invalid test Case 3:**  Input:  Enter the value of x and y:  0  0  Output:  All zeros |
| 1. We can also overload a unary operator in C++ by using a friend function. Write a C++ program to overloaded ++ operator relative to the Test class using a member function.   **[Module 7, Easy]** |
| Case 1:  Input:  12, 22, 33  Output:  12, 22, 33  13, 23, 34  14, 24, 35  Case 2:  Input:  10, 21, 12  Output:  10, 21, 12  11, 22, 13  12, 23, 14  **Invalid test Case 3:**  Input:  0, 0, 0  Output:  All zeros |
| * + 1. Write a C++ program showing runtime behaviour of virtual functions with following conditions  1. Virtual functions cannot be static. 2. A virtual function can be a friend function of another class. 3. Virtual functions should be accessed using pointer or reference of base class type to achieve runtime polymorphism. 4. The prototype of virtual functions should be the same in the base as well as derived class. 5. They are always defined in the base class and overridden in a derived class. It is not mandatory for the derived class to override (or re-define the virtual function), in that case, the base class version of the function is used. 6. A class may have virtual destructor but it cannot have a virtual constructor.   **[Module 7, Hard]**  Case1:  Input (predefined input)  Output:  print derived class  show base class  Case1:  Input:  Static Employee No  18982  Employee name  Gobinath  Salary  60000  Output:  Employee No: 18982  Employee name: Gobinath  Salary: 60000  Valid Pay Scale  Case 2:  Input  Employee No: 12928  Employee name: Arun  Salary: 20000  Output:  Employee No: 18982  Employee name: Gobinath  Invalid Employee  Minimum Pay Scale for employee is 40000 |