C-DAC MUMBAI

Object Oriented Programming using C++

Assigment-3

Q1. Create a class Employee with:

- int id
- string name
- mutable int accessCount
- A **const member function** display() that increments accessCount
- A **user-defined copy constructor** (deep copy where applicable)
- Show difference between:
    - Shallow copy
    - Deep copy
    - Default copy constructor
    - Your user-defined copy constructor

Tasks:

1. Create object e1, call display() multiple times.
2. Create e2 = e1; and prove whether copying was shallow or deep.
3. Explain why display() must be const and why accessCount must be mutable.

Q2. Create a class Student with:

- roll, name, marks
- Parameterized constructor
- Destructor that prints "Destroying student …"

Tasks:

1. Create an **array of 3 Student objects**.
2. Write all student details to a file "students.txt".
3. Read the file again and print contents.
4. Explain constructor/destructor call sequence for array of objects.

Q3. Create a Matrix class using dynamic 2D allocation.

Implement the following:

- operator>> to **input** matrix
- operator<< to **display** matrix
- operator+ for matrix addition
- operator== to check equality

Tasks:

1. Create two matrices using cin >> m1 >> m2;
2. Add them: m3 = m1 + m2;
3. Compare matrices using if(m1 == m2)
4. Deallocate memory in destructor.

Q4. Write a program to divide two integers.

Requirements:

1. Throw a **custom exception** DivideByZero when denominator = 0.
2. Use **nested try-catch** where:
    - Inner catch prints "Inside inner catch"
    - Rethrows the exception
    - Outer catch prints "Handled in outer catch"
3. Use an **exception specification list** on a function that may throw only DivideByZero.

Q5. Create a class Number with:

- Private int value
- A **friend function** to compare two objects (operator>)
- A **friend class** Inspector that can read private data
- Overload:
    - Pre-increment ++n
    - Post-increment n++
    - Assignment operator =

Tasks:

1. Show difference between pre & post increment.
2. Use Inspector to print private members.
3. Compare objects using friend function.

Q6. Create a Vector class (not std::vector) with:

- A dynamic int array
- Size variable
- Overloaded operator[] to access elements
- Overloaded operator() to return sum of all elements

Tasks:

1. Initialize an object using **aggregate initialization** (ONLY if suitable).
2. Demonstrate:
3. v[2] = 50;
4. cout << v( ); // should compute sum
5. Explain difference between index and function-call operators.

Q7. Implement your own String class (char pointer) with:

- Constructor
- Copy constructor
- Destructor
- Assignment operator
- operator+ for concatenation
- operator<< / operator>>

Tasks:

1. Input two strings using your class.
2. Concatenate them using s3 = s1 + s2;
3. Print all three strings.
4. Demonstrate memory handling (deep copy)