# OOPS_ST3_Practice / Sample Codes

By - Harshpreet Singh, 2210990393, +91-9056299166

## Question 1

Hours, minutes, and seconds (HH:MM:SS) should be the three data members of the class Time. Use the overload + operator to add two more time objects. The first line of input comprises hours, minutes, and seconds (separated by spaces). The second line provides the same information again, again separated by spaces: hours, minutes, and seconds. Your job is to sum these two times and output the result.
Input Format:
First line : hours , minutes and seconds (space separated)
Second line: hours , minutes and seconds (space separated)
Output Format:
Addition of two time

```
//
// Created by HARSHPREET SINGH on 30.11.2023.
//
#include <bits/stdc++.h>
using namespace std;

class Time{
public:
    int hr;
    int min;
    int sec;

    Time(int h, int m, int s){
        hr = h; min = m; sec = s;
    }
```

```
    void display(){
        cout << hr  << " : "<< min <<" : "<< sec;
    }

    Time operator+(Time inp){
        int rhr; int rmin; int rsec;

        rsec = sec + inp.sec;
        rmin = min + inp.min + (rsec / 60);
        rhr = hr + inp.hr + (rmin / 60);

        rsec %= 60;
        rmin %= 60;
        rhr %= 24;

        return Time(rhr,rmin,rsec);
    }
};

int main(){
    int h1, m1, s1; int h2,m2,s2;
    cin >> h1 >> m1 >> s1; cin >> h2 >> m2 >> s2;
    Time t1(h1,m1,s1);
    Time t2(h2,m2,s2);

    Time t3 = t1 + t2;
    t3.display();
    return 0;
}
```

## Question 2

Create a matrix class to represent two-dimensional arrays. To add two 2D arrays, use the overloading of the + operator. To accomplish addition, include the overloaded operators and required member functions. Give a succinct implementation of how it's done.
Input:
First line contains number of rows ‘M’ and number of columns ‘N’
Second line contains MxN elements on first matrix
Third line contains MxN elements on second matrix
Output:
Addition of two matrix after operator overloading

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>
using namespace std;

class Matrix {
```

```
private:
    int **mat;
    int rows;
    int cols;

public:
    Matrix(int m, int n) {
        rows = m;
        cols = n;
        mat = new int*[rows];
        for (int i = 0; i < rows; ++i) {
            mat[i] = new int[cols];
        }
    }

    void inputMatrix() {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cin >> mat[i][j];
            }
        }
    }

    Matrix operator+(Matrix other) {
        if (rows != other.rows || cols != other.cols) {
            cerr << "Matrices have different dimensions, addition not possible." << endl;
            return Matrix(0, 0);
        }

        Matrix result(rows, cols);
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                result.mat[i][j] = mat[i][j] + other.mat[i][j];
            }
        }
        return result;
    }

    void displayMatrix() const {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cout << mat[i][j] << " ";
            }
            cout << endl;
        }
    }
};

int main() {
    int m, n;
    cin >> m >> n;

    Matrix matrix1(m, n);
    matrix1.inputMatrix();

    Matrix matrix2(m, n);
    matrix2.inputMatrix();

    Matrix result = matrix1 + matrix2;
```

```
    cout << "Addition of two matrices:" << endl;
    result.displayMatrix();

    return 0;
}
```

## Question 3

Create a C++ class called ArraySorter, which has a constructor, destructor, and member function called sortArray. This function uses the Bubble Sort algorithm to sort an array of numbers in ascending order. The class member variable should be initialised by the constructor using an integer array that is passed in as input. Any dynamic memory allocation should be cleaned up by the destructor. The Bubble Sort algorithm should be used by the sortArray function to sort the array.
Input:
First line contains length of array
second line contain array elements

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>
using namespace std;

class ArraySorter {
private:
    int *arr;
    int size;

public:
    // Constructor
    ArraySorter(int length){
        size = length;
        arr = new int[size];
    }

    // Destructor
    ~ArraySorter() {
        delete[] arr;
    }

    // Function to input array elements
    void inputArray() {
        for (int i = 0; i < size; ++i) {
            cin >> arr[i];
        }
    }

    // Function to perform Bubble Sort
    void sortArray() {
        for (int i = 0; i < size - 1; ++i) {
```

```
            for (int j = 0; j < size - i - 1; ++j) {
                if (arr[j] > arr[j + 1]) {
                    // Swapping elements
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }


    // Function to display sorted array
    void displayArray() {
        for (int i = 0; i < size; ++i) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

int main() {
    int length;
    cin >> length;

    ArraySorter sorter(length);
    sorter.inputArray();

    sorter.sortArray();

    cout << "Sorted Array: ";
    sorter.displayArray();

    return 0;
}
```

## Question 4

You are required to design a C++ program for managing a library system. Implement a class Book to represent a book with the following attributes: title, author, and publicationYear. Q.

Implement the necessary constructor and destructor for the Book class. The constructor should initialize the attributes, and the destructor should display a message indicating the destruction of the object.

Implement the class and demonstrate the usage of the constructor and destructor by creating objects and observing their creation and destruction.

Implement the following class:

**Book Class:**

Properties:  title (string), author (string), publicationYear (integer)

Methods:

**Book(string title, string author, int publicationYear)** - A constructor to initialize the title, author, and publicationYear of the book.

Destructor - Display a message indicating the destruction of the object.

**Input:**

First line contains books name

second line contains author name

Third line contains  year of publication

**Example Test case:**

| Input | Output |
|---|---|
| Harry Potter J.K. Rowling 1965 | Book Created: Harry Potter Book author: J.K. Rowling Destruction of Book: Harry Potter |

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>
#include <string>
#include <cstring>

using namespace std;

class Book {
private:
    string title;
    string author;
    int publicationYear;

public:
    // Constructor
    Book(string t, string a, int year){
        title = t;
        author = a;
        publicationYear = year;
        cout << "Book Created: " << title << endl;
        cout << "Book author: " << author << endl;
    }

    // Destructor
    ~Book() {
        cout << "Destruction of Book: " << title << endl;
    }
};

int main() {
    string bookTitle, authorName;
```

```
    int year;

    // Input
    getline(cin, bookTitle);
    getline(cin, authorName);
    cin >> year;
    cin.ignore(); // Ignore newline character

    // Creating object of Book class
    Book bookObj(bookTitle, authorName, year);

    return 0;
}
```

## Question 5

> Creating a class named Matrix that represents a two-dimensional integer matrix is the assignment assigned to you. It is your responsibility to overuse the * operator to multiply two matrices and the + operator to add two matrices. You also need to write a printMatrix function that outputs the matrix's elements.
> Print the + operator result on two matrices at the end, followed by the * operator result, separated by an empty line.

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>
using namespace std;

class Matrix {
private:
    int **mat;
    int rows;
    int cols;

public:
    Matrix(int m, int n) {
        rows = m;
        cols = n;
        mat = new int*[rows];
        for (int i = 0; i < rows; ++i) {
            mat[i] = new int[cols];
        }
    }
```

```
    void inputMatrix() {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cin >> mat[i][j];
            }
        }
    }

    Matrix operator+(Matrix other) {
        if (rows != other.rows || cols != other.cols) {
            cerr << "Matrices have different dimensions, addition not possible." << endl;
            return Matrix(0, 0);
        }

        Matrix result(rows, cols);
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                result.mat[i][j] = mat[i][j] + other.mat[i][j];
            }
        }
        return result;
    }

    Matrix operator*(const Matrix& other) const {
        if (cols != other.rows) {
            cerr << "Matrix multiplication not possible due to mismatched dimensions." << endl;
            return Matrix(0, 0);
        }

        Matrix result(rows, other.cols);
        for (int i = 0; i < rows; ++i)
            for (int j = 0; j < other.cols; ++j)
                for (int k = 0; k < cols; ++k)
                    result.mat[i][j] += mat[i][k] * other.mat[k][j];

        return result;
    }

    void displayMatrix() const {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cout << mat[i][j] << " ";
            }
            cout << endl;
        }
    }
};

int main() {
    int m, n;
    cin >> m >> n;

    Matrix matrix1(m, n);
    matrix1.inputMatrix();

    Matrix matrix2(m, n);
    matrix2.inputMatrix();
```

```
    Matrix result = matrix1 + matrix2;
    Matrix result2 = matrix1 * matrix2;

    cout << "Addition of two matrices:" << endl;
    result.displayMatrix();

    cout << "Multiplication of 2 Matrices:" << endl;
    result2.displayMatrix();

    return 0;
}
```

## Question 6

Two strings, String1 and String2, are provided to you.
Create the StringManipulator C++ class, which has the following features:
(i) To compare two instances of StringManipulator, overload the > operator. If String1 is lexicographically larger than String2, it should return true; if not, it should return false.
(ii) To compare two instances of StringManipulator, overload the < operator. If String1 is lexicographically smaller than String2, it should return true; if not, it should return false.
(iii) To compare two instances of StringManipulator, overload the == operator. If String1 and String2 are equal, it should return true; if not, it should return false.After completing each process, print the outcome on a different line.

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>
#include <string>

using namespace std;

class StringManipulator {
private:
    string str;

public:
    StringManipulator(string s){
        str = s;
    }

    bool operator>(StringManipulator other){
        return str > other.str;
    }

    bool operator<(StringManipulator other){
        return str < other.str;
    }
```

```
    bool operator==(StringManipulator other){
        return str == other.str;
    }
};

int main() {
    string str1, str2;
    cin >> str1 >> str2;

    StringManipulator sm1(str1);
    StringManipulator sm2(str2);

    string g,s,e;
    (sm1 > sm2) ? g = "true" : g = "false";
    (sm1 < sm2) ? s = "true" : s = "false";
    (sm1 == sm2) ? e = "true" : e = "false";
    cout << "String1 > String2: " << g << endl;
    cout << "String1 < String2: " << s << endl;
    cout << "String1 == String2: " << e << endl;

    return 0;
}
```

## Question 7

Create a class Shape with method getArea(), inherit two classes: Square and Circle from Shape and override the getArea() method from Shape in Square and Circle to Calculate the area of Circle and Square by creating
an instance of both Square and Circle.

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>

using namespace std;

class Shape {
public:
    virtual double getArea() const = 0;
};

class Square : public Shape {
private:
    double side;

public:
    Square(double s){
        side = s;
```

```
    }

    double getArea() const override {
        return side * side;
    }
};

class Circle : public Shape {
private:
    double radius;

public:
    Circle(double r){
        radius = r;
    }

    double getArea() const override {
        return 3.14159 * radius * radius; // Using Pi as approximately 3.14159
    }
};

int main() {
    // Creating instances of Square and Circle
    Square square(5.0); // Square with side 5 units
    Circle circle(3.0); // Circle with radius 3 units

    // Calculating and displaying areas
    cout << "Area of Square: " << square.getArea() << endl;
    cout << "Area of Circle: " << circle.getArea() << endl;

    return 0;
}
```

## Question 8

create a class Complex , overload the operators ++ , – (both pre increment and post increment) which increments and decrements the values of real and imag (both) accordingly.
Finally display the Complex number after performing post increment and pre decrement in the format : real + i imag

```
//
// Created by HARSHPREET SINGH on 30.11.2023.
//
#include <bits/stdc++.h>
using namespace std;

class Complex{
public:
    int real;
    int img;
```

```
    Complex(int r, int i){
        real = r;
        img = i;
    }
    Complex(int r){
        real = r;
        img = 0;
    }
    Complex operator-(){
        return Complex(-real,-img);
    }

    Complex operator++(){ // preincrement
        return Complex(++real,++img);
    }
    Complex operator++(int){ // postincrement
        return Complex(++real,++img);
    }
    Complex operator--(){
        return Complex(--real,--img);
    }
    void display(){
        cout << real << " + i" << img << endl;
    }
};

int main(){
    Complex c1(1,1);
    Complex c2(2,2);
    c1.display();
    Complex negC1 = -c1;
    negC1.display();
    Complex plusC1 = ++c1;
    plusC1.display();
    Complex minusC1 = --c1;
    minusC1.display();
    return 0;
}
```

## Question 9

Make a class named Fruit with a data member to calculate the number of fruits in a basket. Create two other class named Apples and Mangoes to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket.

```
//
// Created by HARSHPREET SINGH on 30/11/2023.
//
#include <iostream>

// Base class Fruit
```

```
class Fruit {
protected:
    int numFruits;

public:
    Fruit(){
        numFruits = 0;
    }

    void addToBasket(int count) {
        numFruits += count;
    }

    int getNumFruits() const{
        return numFruits;
    }
};

// Derived class Apples
class Apples : public Fruit {
public:
    void addApples(int count) {
        addToBasket(count);
    }

    int getApples() const {
        return getNumFruits();
    }
};

// Derived class Mangoes
class Mangoes : public Fruit {
public:
    void addMangoes(int count) {
        addToBasket(count);
    }

    int getMangoes() const {
        return getNumFruits();
    }
};

int main() {
    using namespace std;

    Apples appleBasket;
    Mangoes mangoBasket;

    appleBasket.addApples(5); // Adding 5 apples to the basket
    mangoBasket.addMangoes(8); // Adding 8 mangoes to the basket

    // Display the count of each type of fruit and the total number of fruits
    cout << "Number of Apples: " << appleBasket.getApples() << endl;
    cout << "Number of Mangoes: " << mangoBasket.getMangoes() << endl;
    cout << "Total number of Fruits: " << appleBasket.getNumFruits() + mangoBasket.getNumFruits() << endl;

    return 0;
}
```

## Question 10

Create two classes named Mammals and MarineAnimals. Create another class named BlueWhale which inherits both the above classes. Now, create a function in each of these
classes which prints "I am mammal", "I am a marine animal" and "I belong to both the categories: Mammals as well as Marine Animals" respectively.
Now, create an object for each of the above class

```
#include <iostream>

using namespace std;

// Base class Mammals
class Mammals {
public:
    void displayMammal() {
        cout << "I am a mammal." << endl;
    }
};

// Base class MarineAnimals
class MarineAnimals {
public:
    void displayMarineAnimal() {
        cout << "I am a marine animal." << endl;
    }
};

// Derived class BlueWhale inheriting from Mammals and MarineAnimals
class BlueWhale : public Mammals, public MarineAnimals {
public:
    void displayBlueWhale() {
        cout << "I belong to both the categories: Mammals as well as Marine Animals." << endl;
    }
};

int main() {
    Mammals mammal;
    MarineAnimals marineAnimal;
    BlueWhale blueWhale;

    cout << "Calling functions using objects of respective classes:" << endl;
    cout << "\nFunction of Mammals using object of Mammals:" << endl;
    mammal.displayMammal();

    cout << "\nFunction of MarineAnimals using object of MarineAnimals:" << endl;
    marineAnimal.displayMarineAnimal();

    cout << "\nFunction of BlueWhale using object of BlueWhale:" << endl;
    blueWhale.displayBlueWhale();

    cout << "\nFunctions of each parent class using object of BlueWhale:" << endl;
    blueWhale.displayMammal(); // Function of Mammals through BlueWhale object
```

```
    blueWhale.displayMarineAnimal(); // Function of MarineAnimals through BlueWhale object

    return 0;
}
```

The END.

If you made it till here… All The Best!!!

if you wanna check me out → InstaID i am quite active lol → harshpreet0402