**LAB ASSIGNMENT**

1. Design a class Matrix of dimension 3x3. Overload + operator to find sum of two matrices.

**Source Code:** #include <iostream> using namespace std;

class Matrix { private:

int mat[3][3]; public: Matrix() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) mat[i][j] = 0;

}

void input() { cout << "Enter elements of 3x3 matrix:\n"; for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) { cin >> mat[i][j];

}

}

}

void display() const { for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) { cout << mat[i][j] << " ";

} cout << endl;

}

}

Matrix operator+(const Matrix& *m*) const { Matrix temp; for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) {

temp.mat[i][j] = mat[i][j] + m.mat[i][j];

}

} return temp;

}

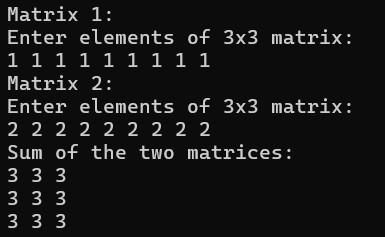
};

int main() { Matrix m1, m2, sum; cout << "Matrix 1:"<<endl; m1.input(); cout << "Matrix 2:"<<endl; m2.input(); sum = m1 + m2; cout << "Sum of the two matrices:"<<endl; sum.display();

return 0;

}

**Output**



2. Define a class string and use + and > operators to concatenate and compare two strings respectively.

**Source Code:**

#include <iostream>

#include <cstring> #include <string> using namespace std; class String { string str; public:

void input() { cout << "Enter String: "; cin >> str;

} void display() { cout << str;

}

String operator+( String& *s*) { String temp; temp.str = str + s.str; return temp;

}

bool operator>(String& *s*) {

return str > s.str;

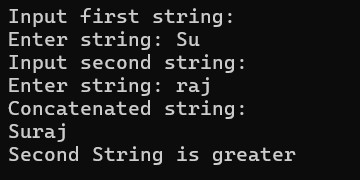
}

};

int main() { String s1, s2, s3; cout << "Input first String:"<<endl; s1.input(); cout << "Input second String:"<<endl; s2.input(); s3 = s1 + s2; cout << "Concatenated String: "<<endl; s3.display(); cout << endl; if (s1 > s2) cout << "First String is greater." << endl; else cout << "Second String is greater." << endl; return 0;

}

**Output:**



3. Write a program to implement vector addition and subtraction using operator overloading. **Source Code:** #include <iostream> using namespace std; class Vector {

int x, y, z; public: Vector() { x = 0; y = 0; z = 0;

}

Vector(int *a*, int *b*, int *c*) { x = a; y = b; z = c;

} void input() { cin >> x >> y >> z;

} void display() { cout << "(" << x << ", " << y << ", " << z << ")";

}

Vector operator+(const Vector& *v*) const { return Vector(x + v.x, y + v.y, z + v.z);

}

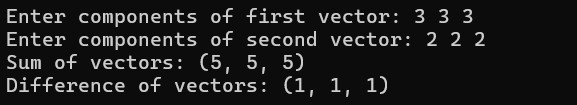
Vector operator-(const Vector& *v*) const { return Vector(x - v.x, y - v.y, z - v.z); }

}; int main() {

Vector v1, v2, sum, diff; cout << "Enter components of first vector: "; v1.input(); cout << "Enter components of second vector: "; v2.input(); sum = v1 + v2; diff = v1 - v2; cout << "Sum of vectors: "; sum.display(); cout << endl; cout << "Difference of vectors: "; diff.display(); cout << endl; return 0;

}

**Output:**



4. Design a class Matrix, overload ++ and -- operator to increment and decrement each element of the matrix by 1.

**Source Code:** #include <iostream> using namespace std; class Matrix { int mat[3][3]; public: Matrix() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) mat[i][j] = 0;

} void input() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) cin >> mat[i][j];

} void display() { for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) cout << mat[i][j] << " "; cout << endl;

}

}

Matrix& operator++() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) ++mat[i][j]; return \*this;

}

Matrix operator++(int) {

Matrix temp = \*this; ++(\*this); return temp;

}

Matrix& operator--() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) --mat[i][j]; return \*this;

}

Matrix operator--(int) {

Matrix temp = \*this; --(\*this); return temp; }

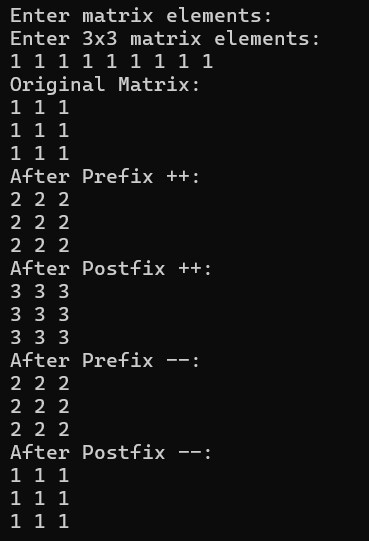
}; int main() { Matrix m, result; cout << "Enter elements of 3x3 matrix:" << endl;

m.input(); cout << "Original matrix:" << endl;

m.display(); result = ++m; cout << "Result of pre-increment :" << endl; result.display(); result = m++; cout << "Result of post-increment " << endl; result.display(); result = --m; cout << "Result of pre-decrement " << endl; result.display(); result = m--; cout << "Result of post-decrement " << endl; result.display(); return 0;

}

**Output:**



5. Write a program to access elements of a vector class with index operator.

**Source Code:** #include <iostream> using namespace std; class Vector { int arr[10]; int size; public:

Vector(int *s* = 0) { size = s; for(int i = 0; i < 10; ++i) arr[i] = 0;

} void input() { cout << "Enter size of vector (max 10): "; cin >> size; cout << "Enter elements: "; for(int i = 0; i < size; ++i) cin >> arr[i];

}

int& operator[](int *index*) { return arr[index];

}

void display() { for(int i = 0; i < size; ++i) cout << arr[i] << " "; cout << endl;

}

}; int main() {

Vector v;

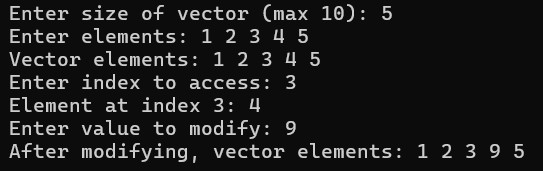
v.input(); cout << "Vector elements: ";

v.display(); int idx,temp; cout << "Enter index to access: "; cin >> idx; cout << "Element at index " << idx << ": " << v[idx] << endl; cout << "Enter value to modify: "; cin >> temp; v[idx] = temp; cout << "After modifying, vector elements: ";

v.display(); return 0;

}

**Output:**



6. Write a program to multiply two matrices by overloading the \* operator.

**Source Code:** #include <iostream> using namespace std; class Matrix { int mat[3][3]; public: Matrix() { for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) mat[i][j] = 0;

} void input() { cout << "Enter elements of 3x3 matrix:\n"; for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) cin >> mat[i][j];

} void display() { for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) cout << mat[i][j] << " "; cout << endl;

}

}

Matrix operator\*(const Matrix& *m*) const {

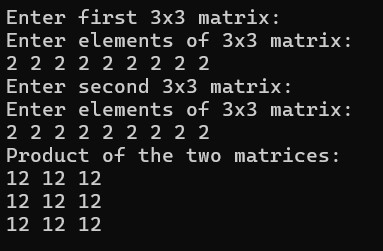
Matrix result; for(int i=0; i<3; ++i) { for(int j=0; j<3; ++j) { result.mat[i][j] = 0; for(int k=0; k<3; ++k) { result.mat[i][j] += mat[i][k] \* m.mat[k][j];

} } } return result;

} }; int main() { Matrix m1, m2, product; cout << "Enter first 3x3 matrix:" << endl; m1.input(); cout << "Enter second 3x3 matrix:" << endl; m2.input(); product = m1 \* m2; cout << "Product of the two matrices:" << endl; product.display(); return 0;

}

**Output:**



7. Create a class named City that will have two member variables CityName and DistFromKtm (float). Add member functions to set and retrieve the CityName and

DistFromKtm separately. Add operator overloading to find the distance between the cities

(just find the difference of DistFromKtm) and sum of distance of those cities from

Kathmandu. In the main function,initialize three city objects. Set the first and second city to be Pokhara and Dhangadi. Display the sum of DistFromKtm of Pokhara and Dhangadi and distance between Pokhara and Dhangadi.

**Source Code:**

#include <iostream>

#include <cmath> #include <string> using namespace std;

class City { private:

string CityName; float DistFromKtm; public:

void setCityName(string *name*) {

CityName = name;

}

void setDistFromKtm(float *dist*) {

DistFromKtm = dist;

}

string getCityName() { return CityName;

}

float getDistFromKtm() { return DistFromKtm;

} float operator-(City& *c*) { return fabs(DistFromKtm - c.DistFromKtm);

}

float operator+(City& *c*) { return DistFromKtm + c.DistFromKtm;

}

};

int main() {

City city1, city2, city3; city1.setCityName("Pokhara"); city1.setDistFromKtm(200.0f); city2.setCityName("Dhangadi"); city2.setDistFromKtm(660.0f);

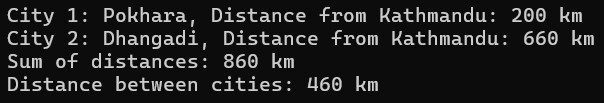
cout << "City 1: " << city1.getCityName() << ", Distance from Kathmandu: " << city1.getDistFromKtm() << " km" << endl;

cout << "City 2: " << city2.getCityName() << ", Distance from Kathmandu: " << city2.getDistFromKtm() << " km" << endl; city3.setCityName("Sum of distances"); city3.setDistFromKtm(city1 + city2); cout << city3.getCityName() << ": " << city3.getDistFromKtm() << " km" << endl; city3.setCityName("Distance between cities"); city3.setDistFromKtm(city1 - city2); cout << city3.getCityName() << ": " << city3.getDistFromKtm() << " km" << endl;

return 0;

}

**Output:**



8. Write a program to overload the relational operators to compare the length (in meter and centimeter) of two objects.

**Source Code:** #include <iostream> using namespace std; class Length { private:

int meter; int centimeter; int totalCentimeter() { return meter \* 100 + centimeter;

}

public:

Length(int *m* = 0, int *cm* = 0) { meter = m + cm / 100; centimeter = cm % 100;

} void input() { cout << "Enter length (meter and centimeter): "; cin >> meter >> centimeter; meter += centimeter / 100;

centimeter = centimeter % 100;

} void display() { cout << meter << " meter " << centimeter << " centimeter";

}

bool operator==(Length& *l*) { return totalCentimeter() == l.totalCentimeter();

}

bool operator<(Length& *l*) { return totalCentimeter() < l.totalCentimeter();

}

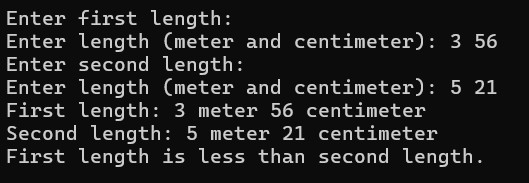
bool operator>(Length& *l*) { return totalCentimeter() > l.totalCentimeter();

} }; int main() { Length l1, l2; cout << "Enter first length:\n"; l1.input(); cout << "Enter second length:\n"; l2.input(); cout << "First length: "; l1.display(); cout << endl; cout << "Second length: "; l2.display(); cout << endl;

if (l1 == l2) cout << "Both lengths are equal." << endl; if (l1 > l2) cout << "First length is greater than second length." << endl; if (l1 < l2) cout << "First length is less than second length." << endl; return 0;

}

**Output:**



9. Write operator functions as member function of a class to overload arithmetic operator +, logical operator <=, and stream operator << to operate on the objects of user-defined type time (hr, min,sec).

**Source Code:** #include <iostream> using namespace std; class Time { int hr, min, sec; void normalize() { int total = hr \* 3600 + min \* 60 + sec; hr = total / 3600;

min = (total % 3600) / 60; sec = total % 60;

}

public:

Time(int h = 0, int m = 0, int s = 0) { hr = h; min = m; sec = s; normalize();

} void input() { cout << "Enter time (hr min sec): "; cin >> hr >> min >> sec; normalize();

}

int toSeconds() const { return hr \* 3600 + min \* 60 + sec;

}

Time operator+(Time t) { int total = toSeconds() + t.toSeconds(); return Time(0, 0, total);

}

int operator<(Time t) { return toSeconds() < t.toSeconds();

}

int operator>(Time t) { return toSeconds() > t.toSeconds(); }

int operator==(Time t) { return toSeconds() == t.toSeconds();

}

friend ostream& operator<<(ostream& out, const Time& t);

};

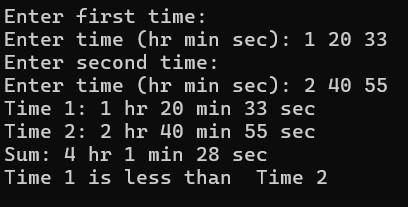
ostream& operator<<(ostream& out, const Time& t) { out << t.hr << " hr " << t.min << " min " << t.sec << " sec"; return out; } int main() { Time t1, t2, t3; cout << "Enter first time:" << endl; t1.input(); cout << "Enter second time:" << endl; t2.input(); t3 = t1 + t2; cout << "Time 1: " << t1 << endl; cout << "Time 2: " << t2 << endl; cout << "Sum: " << t3 << endl;

if (t1 < t2) cout << "Time 1 is less than Time 2" << endl; else if (t1 > t2) cout << "Time 1 is greater than Time 2" << endl;

else cout << "Time 1 is equal to Time 2" << endl; return 0;

}

**Output:**



10. Create a class called time that has separate int member data for hours, minutes, and seconds. One constructor should initialize this data to zero (0), and another should initialize it to fixed values. A member function should display it in 10:45:30 format. The final member function should add two objects of type time passed as arguments using operator overloading.

**Source Code:** #include <iostream> using namespace std; class Time { private:

int hr, min, sec; public: Time() { hr = min = sec = 0;

}

Time(int h, int m, int s) { hr = h;

min = m; sec = s; normalize();

}

void normalize() { min += sec / 60; sec %= 60; hr += min / 60; min %= 60;

}

void display() const {

cout << (hr < 10 ? "0" : "") << hr << ":"<< (min < 10 ? "0" : "") << min << ":"<< (sec < 10 ? "0" : "") << sec << endl;

}

Time operator+(const Time& t) { Time result; result.hr = hr + t.hr; result.min = min + t.min; result.sec = sec + t.sec; result.normalize(); return result;

} }; int main() { Time t1, t2; int h, m, s; cout << "Enter time 1 (hr min sec): "; cin >> h >> m >> s; t1 = Time(h, m, s); cout << "Enter time 2 (hr min sec): "; cin >> h >> m >> s; t2 = Time(h, m, s); Time t3; t3 = t1 + t2; cout << "Time 1: "; t1.display(); cout << "Time 2: "; t2.display(); cout << "Sum: "; t3.display(); return 0;

}

**Output:**

