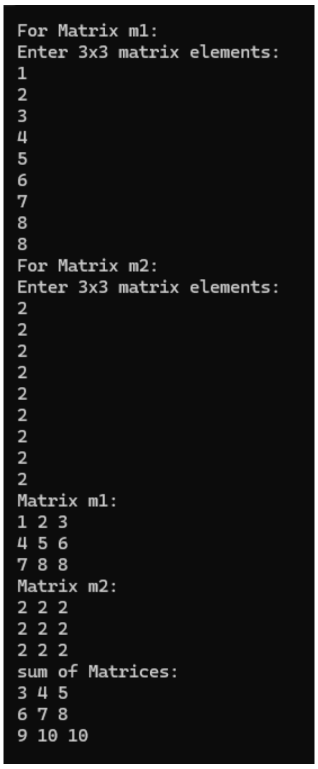
Lab Questions:

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

```
#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 3x3 matrix elements:\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) {
Matrix result;
for (int i = 0; i < 3; i++)
for (int j = 0; j < 3; j++)
result.mat[i][j] = mat[i][j] + m.mat[i][j];
return result;
}
};
int main() {
Matrix m1, m2, m3;
cout << "For Matrix 1:\n";
m1.input();
cout << "For Matrix 2:\n";
m2.input();
cout << "Matrix 1:\n";
m1.display();
cout << "Matrix 2:\n";
m2.display();
m3 = m1 + m2;
cout << "Sum of Matrices:\n";
m3.display();
return 0;
}
```



```
For Matrix m1:
Enter 3x3 matrix elements:
1
2
3
4
5
6
7
8
8
For Matrix m2:
Enter 3x3 matrix elements:
2
2
2
2
2
2
2
2
2
Matrix m1:
1 2 3
4 5 6
7 8 8
Matrix m2:
2 2 2
2 2 2
2 2 2
sum of Matrices:
3 4 5
6 7 8
9 10 10
```

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

```
#include <iostream>
#include <cstring>
using namespace std;
class String {
private:
char str[100];
public:
String() {
str[0] = '\0';
}
String(const char* s) {
strcpy(str, s);
}
void input() {
cout << "Enter string: ";
cin >> str;
}
void display() {
cout << str << endl;
}
String operator+(const String& s) {
String result;
strcpy(result.str, str);
strcat(result.str, s.str);
return result;
}
bool operator>(const String& s) {
return strcmp(str, s.str) > 0;
}
};
int main() {
String s1, s2, s3;
cout << "For String 1:\n";
s1.input();
cout << "For String 2:\n";
s2.input();
cout << "String 1: ";
s1.display();
cout << "String 2: ";
```

s2.display();

s3 = s1 + s2;

cout << "Concatenated String: ";

s3.display();

if (s1 > s2)

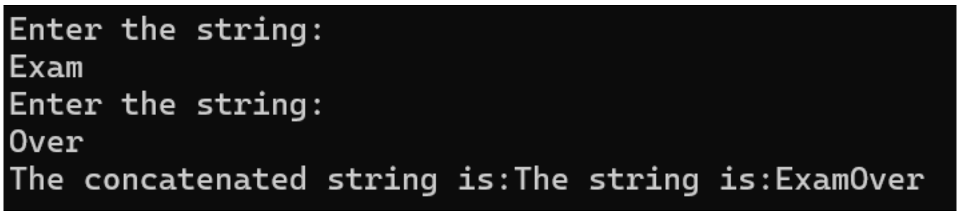
cout << "String 1 is greater than String 2" << endl;

else

cout << "String 1 is not greater than String 2" << endl;

return 0;

}



```
Enter the string:
Exam
Enter the string:
Over
The concatenated string is:The string is:ExamOver
```

3. Write a program to implement vector addition and subtraction using operator overloading.

#include <iostream>

using namespace std;

class Vector {

private:

double x, y, z;

public:

Vector(double xCoord = 0, double yCoord = 0, double zCoord = 0) : x(xCoord), y(yCoord), z(zCoord) {}

void input() {

cout << "Enter x, y, z components: ";

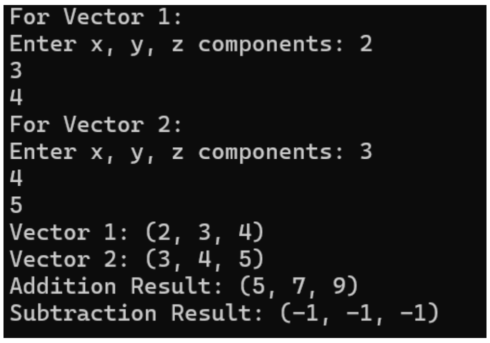
cin >> x >> y >> z;

}

void display() {

cout << "(" << x << ", " << y << ", " << z << ")" << endl;

```
}
Vector operator+(const Vector& v) {
return Vector(x + v.x, y + v.y, z + v.z);
}
Vector operator-(const Vector& v) {
return Vector(x - v.x, y - v.y, z - v.z);
}
};
int main() {
Vector v1, v2, v3, v4;
cout << "For Vector 1:\n";
v1.input();
cout << "For Vector 2:\n";
v2.input();
cout << "Vector 1: ";
v1.display();
cout << "Vector 2: ";
v2.display();
v3 = v1 + v2;
v4 = v1 - v2;
cout << "Addition Result: ";
v3.display();
cout << "Subtraction Result: ";
v4.display();
return 0;
}
```

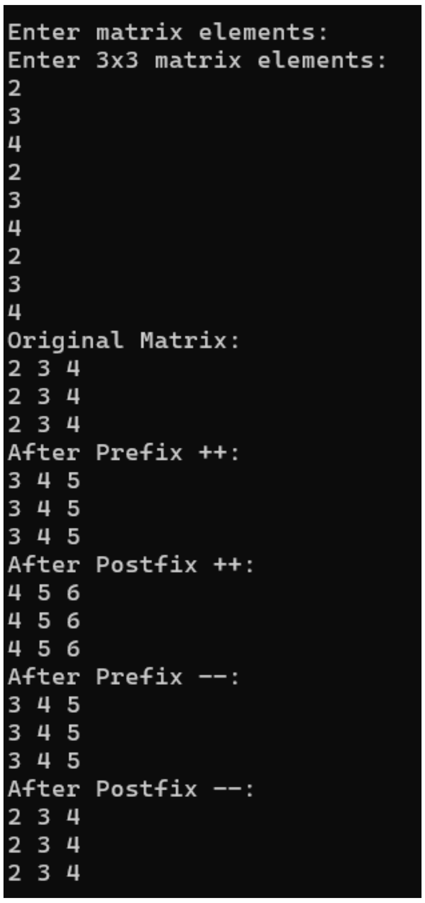


```
For Vector 1:
Enter x, y, z components: 2
3
4
For Vector 2:
Enter x, y, z components: 3
4
5
Vector 1: (2, 3, 4)
Vector 2: (3, 4, 5)
Addition Result: (5, 7, 9)
Subtraction Result: (-1, -1, -1)
```

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

```
#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 3x3 matrix elements:\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++() { // Prefix increment

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

mat[i][j]++;

return *this;
}
Matrix& operator--() { // Prefix decrement
for (int i = 0; i < 3; i++)
for (int j = 0; j < 3; j++)
mat[i][j]--;
return *this;
}
Matrix operator++(int) { // Postfix increment
Matrix temp = *this;
for (int i = 0; i < 3; i++)
for (int j = 0; j < 3; j++)
mat[i][j]++;
return temp;
}
Matrix operator--(int) { // Postfix decrement
Matrix temp = *this;
for (int i = 0; i < 3; i++)
for (int j = 0; j < 3; j++)
mat[i][j]--;
return temp;
}
};
int main() {
Matrix m;
cout << "Enter matrix elements:\n";
m.input();
cout << "Original Matrix:\n";
m.display();
cout << "After Prefix ++:\n";
++m;
```

```
m.display();
cout << "After Postfix ++:\n";
m++;
m.display();
cout << "After Prefix --:\n";
--m;
m.display();
cout << "After Postfix --:\n";
m--;
m.display();
return 0;
}
```

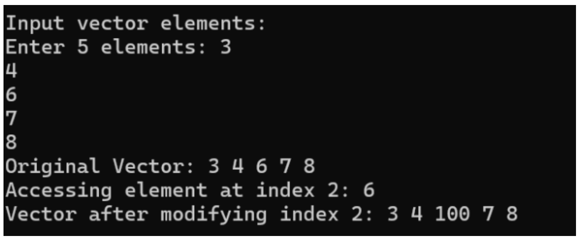


```
Enter matrix elements:
Enter 3x3 matrix elements:
2
3
4
2
3
4
2
3
4
Original Matrix:
2 3 4
2 3 4
2 3 4
After Prefix ++:
3 4 5
3 4 5
3 4 5
After Postfix ++:
4 5 6
4 5 6
4 5 6
After Prefix --:
3 4 5
3 4 5
3 4 5
After Postfix --:
2 3 4
2 3 4
2 3 4
```

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

```
#include <iostream>
using namespace std;
class Vector {
private:
int arr[5]; // Fixed size vector of 5 elements
int size;
public:
Vector() : size(5) {
for (int i = 0; i < size; i++)
arr[i] = 0;
}
void input() {
```

```
cout << "Enter " << size << " elements: ";
for (int i = 0; i < size; i++)
cin >> arr[i];
}
void display() {
for (int i = 0; i < size; i++)
cout << arr[i] << " ";
cout << endl;
}
int& operator[](int index) {
if (index >= 0 && index < size)
return arr[index];
cout << "Index out of bounds!" << endl;
return arr[0]; // Return first element as default for invalid index
}
};
int main() {
Vector v;
cout << "Input vector elements:\n";
v.input();
cout << "Original Vector: ";
v.display();
cout << "Accessing element at index 2: " << v[2] << endl;
v[2] = 100; // Modify element using index operator
cout << "Vector after modifying index 2: ";
v.display();
return 0;
}
```

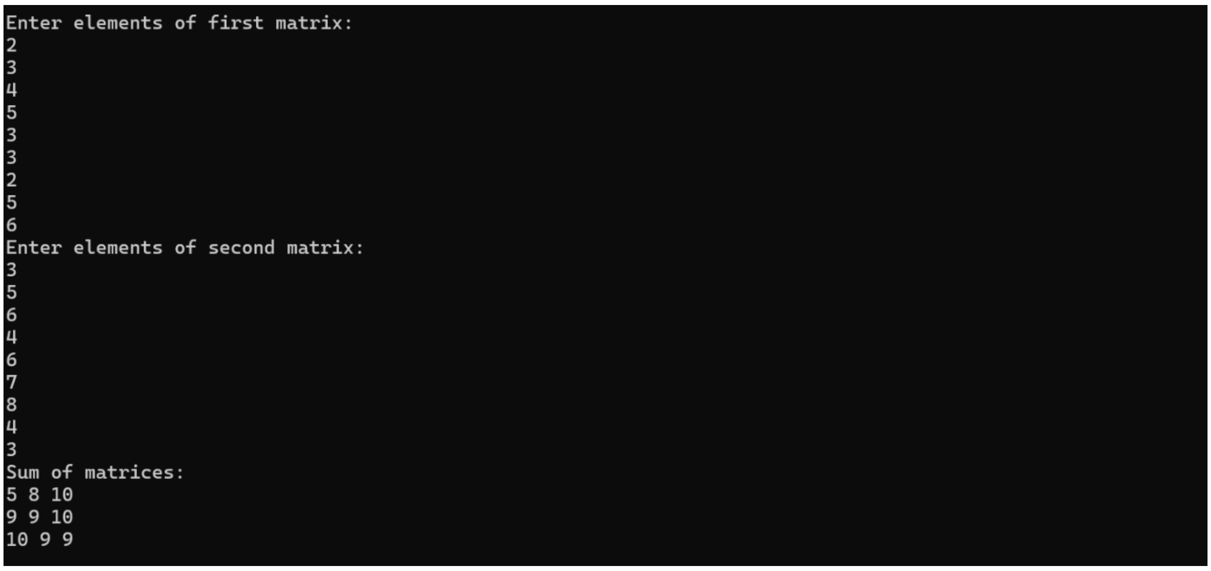


```
Input vector elements:
Enter 5 elements: 3
4
6
7
8
Original Vector: 3 4 6 7 8
Accessing element at index 2: 6
Vector after modifying index 2: 3 4 100 7 8
```

6. Write a program to add two matrices by overloading the + operator.

```
#include <iostream>
using namespace std;
class Matrix {
private:
   int mat[3][3];
public:
   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+(const Matrix& m) {
      Matrix result;
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            result.mat[i][j] = mat[i][j] + m.mat[i][j];
      return result;
   }
};
int main() {
   Matrix m1, m2, sum;
   cout << "Enter elements of first matrix:" << endl;
```

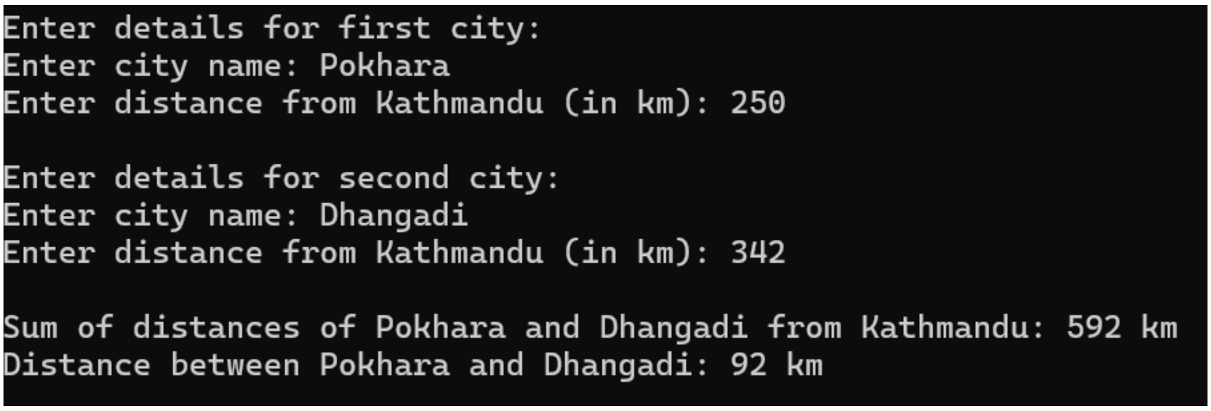
```
    m1.input();

    cout << "Enter elements of second matrix:" << endl;

    m2.input();

    sum = m1 + m2;

    cout << "Sum of matrices:" << endl;

    sum.display();

    return 0;

}
```



```
Enter elements of first matrix:
2
3
4
5
3
3
2
5
6
Enter elements of second matrix:
3
5
6
4
6
7
8
4
3
Sum of matrices:
5 8 10
9 9 10
10 9 9
```

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.

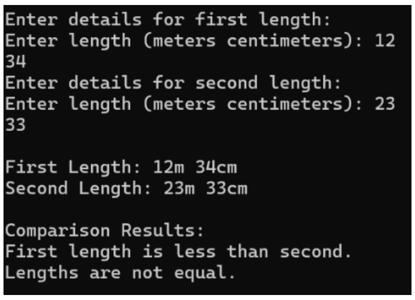
```
#include <iostream>

#include <cmath>

#include <string>

using namespace std;

class City {

private:

    string CityName;
    float DistFromKtm;

public:
    void setCityName(string name) {
        CityName = name;
    }
    void setDistFromKtm(float distance) {
        DistFromKtm = distance;
    }
    string getCityName() const {
        return CityName;
    }
    float getDistFromKtm() const {
        return DistFromKtm;
    }
    float operator-(const City& other) const {
        return fabs(this->DistFromKtm - other.DistFromKtm);
    }
    float operator+(const City& other) const {
        return this->DistFromKtm + other.DistFromKtm;
    }
    void input() {
        cout << "Enter city name: ";
        getline(cin, CityName);
        cout << "Enter distance from Kathmandu (in km): ";
        cin >> DistFromKtm;
        cin.ignore();
    }
};
int main() {
    City city1, city2, city3;

    cout << "Enter details for first city:\n";

    city1.input();

    cout << "\nEnter details for second city:\n";

    city2.input();

    cout << "\nSum of distances of " << city1.getCityName() << " and " <<
city2.getCityName()

        << " from Kathmandu: " << city1 + city2 << " km" << endl;

    cout << "Distance between " << city1.getCityName() << " and " <<
city2.getCityName()

        << ": " << city1 - city2 << " km" << endl;

    return 0;

}
```



```
Enter details for first city:
Enter city name: Pokhara
Enter distance from Kathmandu (in km): 250

Enter details for second city:
Enter city name: Dhangadi
Enter distance from Kathmandu (in km): 342

Sum of distances of Pokhara and Dhangadi from Kathmandu: 592 km
Distance between Pokhara and Dhangadi: 92 km
```

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

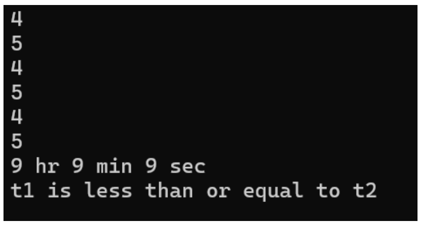
```
#include <iostream>

using namespace std;

class Length {

private:

    int meters;

    int centimeters;


    int toCentimeters() const {
        return meters * 100 + centimeters;
    }
public:
    void setLength(int m, int cm) {
        meters = m;
        centimeters = cm;
    }
    void input() {
        cout << "Enter length (meters centimeters): ";
        cin >> meters >> centimeters;
    }
    void display() const {
        cout << meters << "m " << centimeters << "cm";
    }
    bool operator==(const Length& other) const {
        return this->toCentimeters() == other.toCentimeters();
    }
    bool operator!=(const Length& other) const {
        return this->toCentimeters() != other.toCentimeters();
    }
    bool operator<(const Length& other) const {
        return this->toCentimeters() < other.toCentimeters();
    }

    bool operator>(const Length& other) const {
        return this->toCentimeters() > other.toCentimeters();
    }
    bool operator<=(const Length& other) const {
        return this->toCentimeters() <= other.toCentimeters();
    }
    bool operator>=(const Length& other) const {
        return this->toCentimeters() >= other.toCentimeters();
    }
};
int main() {
    Length l1, l2;
    cout << "Enter details for first length:\n";
    l1.input();
    cout << "Enter details for second length:\n";
    l2.input();
    cout << "\nFirst Length: ";
    l1.display();
    cout << "\nSecond Length: ";
    l2.display();
    cout << "\n\nComparison Results:\n";
    if (l1 == l2)
        cout << "Lengths are equal.\n";
    else if (l1 < l2)
        cout << "First length is less than second.\n";
    else if (l1 > l2)
        cout << "First length is greater than second.\n";
    if (l1 != l2)
        cout << "Lengths are not equal.\n";
    return 0;
}
```



```
Enter details for first length:
Enter length (meters centimeters): 12
34
Enter details for second length:
Enter length (meters centimeters): 23
33

First Length: 12m 34cm
Second Length: 23m 33cm

Comparison Results:
First length is less than second.
Lengths are not equal.
```

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).

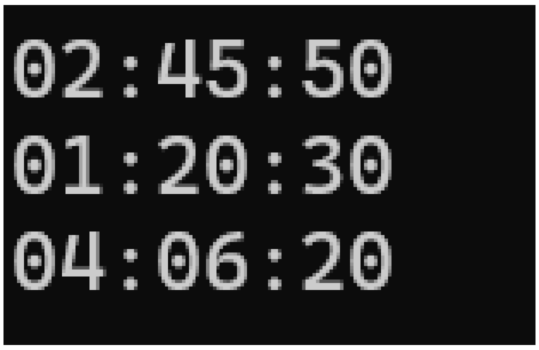
```
#include <iostream>
using namespace std;
class Time {
private:
    int hr, min, sec;
public:
    Time(int h = 0, int m = 0, int s = 0) {
        hr = h;
        min = m;
        sec = s;
        normalize();
    }
    void getInput() {
        cin >> hr >> min >> sec;
        normalize();
    }
    void normalize() {
        min += sec / 60;
        sec = sec % 60;
        hr += min / 60;
        min = min % 60;
    }
    Time operator+(const Time& t) {
        Time temp;
        temp.hr = hr + t.hr;
        temp.min = min + t.min;
        temp.sec = sec + t.sec;
        temp.normalize();
        return temp;
    }
    bool operator<=(const Time& t) {
        if (hr < t.hr) return true;
        if (hr == t.hr && min < t.min) return true;
        if (hr == t.hr && min == t.min && sec <= t.sec) return true;
        return false;
    }
    ostream& operator<<(ostream& out) {
        out << hr << " hr " << min << " min " << sec << " sec";
        return out;
    }
    void display() {
        cout << hr << " hr " << min << " min " << sec << " sec" << endl;
    }
};
int main() {
    Time t1, t2, sum;
    t1.getInput();
    t2.getInput();
    sum = t1 + t2;
    sum.display();
    if (t1 <= t2)
        cout << "t1 is less than or equal to t2" << endl;
    else
        cout << "t1 is greater than t2" << endl;
    return 0;
}
```



```
4
5
4
5
4
5
9 hr 9 min 9 sec
t1 is less than or equal to t2
```

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.

```
#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(2, 45, 50);
    Time t2(1, 20, 30);
    Time t3;
    t3 = t1 + t2;
    t1.display();
    t2.display();
    t3.display();
    return 0;
}
```



```
02:45:50
01:20:30
04:06:20
```