



WORK SHEET 3

Manish Mahato

The British College

Student ID: 23085136



Work Sheet 3

1. Create a `Time` class to store hours and minutes. Implement:

1. Overload the `+` operator to add two `Time` objects
2. Overload the `>` operator to compare two `Time` objects
3. Handle invalid time (`>24` hours or `>60` minutes) by throwing a custom exception

Source code for this:

```
#include <iostream>
using namespace std;

class Time {
    int hours, minutes;

public:
    Time(int h = 0, int m = 0) {
        if (h < 0 || h >= 24 || m < 0 || m >= 60) {
            throw "Invalid time! Hours (0-23), Minutes (0-59)";
        }
        hours = h;
        minutes = m;
    }

    void display() const {
        cout << hours << " hrs " << minutes << " mins\n";
    }

    Time operator+(const Time& t) const {
        int total = (hours + t.hours) * 60 + (minutes + t.minutes);
        return Time((total / 60) % 24, total % 60);
    }

    bool operator>(const Time& t) const {
        return (hours * 60 + minutes) > (t.hours * 60 + t.minutes);
    }
};

int main() {
    try {
        int h1, m1, h2, m2;
        cout << "Enter first time in hour and minute): ";
        cin >> h1 >> m1;
        cout << "Enter second time in hour and minute): ";
        cin >> h2 >> m2;

        Time t1(h1, m1), t2(h2, m2), sum = t1 + t2;

        cout << "First : "; t1.display();
        cout << "Second: "; t2.display();
        cout << "Sum of time: "; sum.display();
    }
```

```

        cout << (t1 > t2 ? "First is greater.\n" : "Second is greater or
equal.\n");

    } catch (const char* msg) {
        cout << "Error: " << msg << endl;
    }

    return 0;
}

```

Output:

```

Run task1_1.cpp x
"/Users/manish/Desktop/4Th C++/worksheet3/worksheet3/task1_1"
Enter first time in hour and minute): 1
24
Enter second time in hour and minute): 4
56
First : 1 hrs 24 mins
Second: 4 hrs 56 mins
Sum of time: 6 hrs 20 mins
Second is greater or equal.

Process finished with exit code 0

```

Task 2: 70 marks

1. Create a base class `Vehicle` and two derived classes `Car` and `Bike`:
 1. Vehicle has registration number and color
 2. Car adds number of seats
 3. Bike adds engine capacity
 4. Each class should have its own method to write its details to a file
 5. Include proper inheritance and method overriding

Source code for this:

```
#include <iostream>
#include <fstream>
using namespace std;

class Vehicle {
protected:
    string regNo, color;
public:
    Vehicle(string r, string c) : regNo(r), color(c) {}

    virtual void writeToFile(ofstream& file) {
        file << "Registration Number: " << regNo << "\nColor: " << color <<
endl;
    }

    virtual ~Vehicle() {}
};

class Car : public Vehicle {
    int seats;
public:
    Car(string r, string c, int s) : Vehicle(r, c), seats(s) {}

    void writeToFile(ofstream& file) override {
        Vehicle::writeToFile(file);
        file << "Seats: " << seats << "\n---\n";
    }
};

class Bike : public Vehicle {
    int engineCC;
public:
    Bike(string r, string c, int cc) : Vehicle(r, c), engineCC(cc) {}

    void writeToFile(ofstream& file) override {
        Vehicle::writeToFile(file);
        file << "Engine: " << engineCC << " CC\n---\n";
    }
};

int main() {
    int choice;
    string reg, color;
    ofstream file("vehicle_details.txt", ios::app);

    if (!file) {
        cout << "File error.\n";
        return 1;
    }

    cout << "1 for Car\n2 for Bike\nChoose: ";
    cin >> choice;
```

```

cin.ignore();

cout << "Registration Number: ";
getline(cin, reg);
cout << "Color: ";
getline(cin, color);

if (choice == 1) {
    int seats;
    cout << "Seating Capacity: ";
    cin >> seats;
    Car c(reg, color, seats);
    c.writeToFile(file);
} else if (choice == 2) {
    int cc;
    cout << "Engine (CC): ";
    cin >> cc;
    Bike b(reg, color, cc);
    b.writeToFile(file);
} else {
    cout << "Invalid choice.\n";
}

cout << "Details saved.\n";
file.close();
return 0;
}

```

Output:

```

Run Task_2_1.cpp x
"/Users/manish/Desktop/Manish/work sheet 3/Task_2_1"
1 for Car
2 for Bike
Choose: 1
Registration Number: 123
Color: red
Seating Capacity: 5
Details saved.

Process finished with exit code 0

```

2. Create a program that:

1. Reads student records (roll, name, marks) from a text file
2. Throws an exception if marks are not between 0 and 100
3. Allows adding new records with proper validation
4. Saves modified records back to file

Source code for this:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

// Base class
class Person {
protected:
    int roll;
    string name;

public:
    Person() : roll(0), name("") {}
    Person(int r, string n) : roll(r), name(n) {}

    virtual void input() {
        cout << "Roll No: ";
        cin >> roll;
        cout << "Name: ";
        cin >> name;
    }

    virtual void display() const {
        cout << roll << " " << name;
    }

    int getRoll() const { return roll; }
    string getName() const { return name; }
};

// Derived class
class Student : public Person {
    int marks;

public:
    Student() : Person(), marks(0) {}
    Student(int r, string n, int m) : Person(r, n), marks(m) {}

    void input() override {
        Person::input();
        cout << "Marks (0-100): ";
        cin >> marks;

        while (marks < 0 || marks > 100) {
```

```

        cout << "Invalid marks! Enter again (0-100): ";
        cin >> marks;
    }
}

void display() const override {
    Person::display();
    cout << " " << marks << endl;
}

int getMarks() const { return marks; }

// File I/O
static void loadFromFile(const string& filename, vector<Student>&
students) {
    ifstream inFile(filename);
    int r, m;
    string n;

    while (inFile >> r >> n >> m) {
        students.push_back(Student(r, n, m));
    }

    inFile.close();
}

static void saveToFile(const string& filename, const vector<Student>&
students) {
    ofstream outFile(filename);
    for (const auto& s : students) {
        outFile << s.getRoll() << " " << s.getName() << " " <<
s.getMarks() << endl;
    }
    outFile.close();
}
};

int main() {
    string filename = "students.txt";
    vector<Student> students;

    Student::loadFromFile(filename, students);

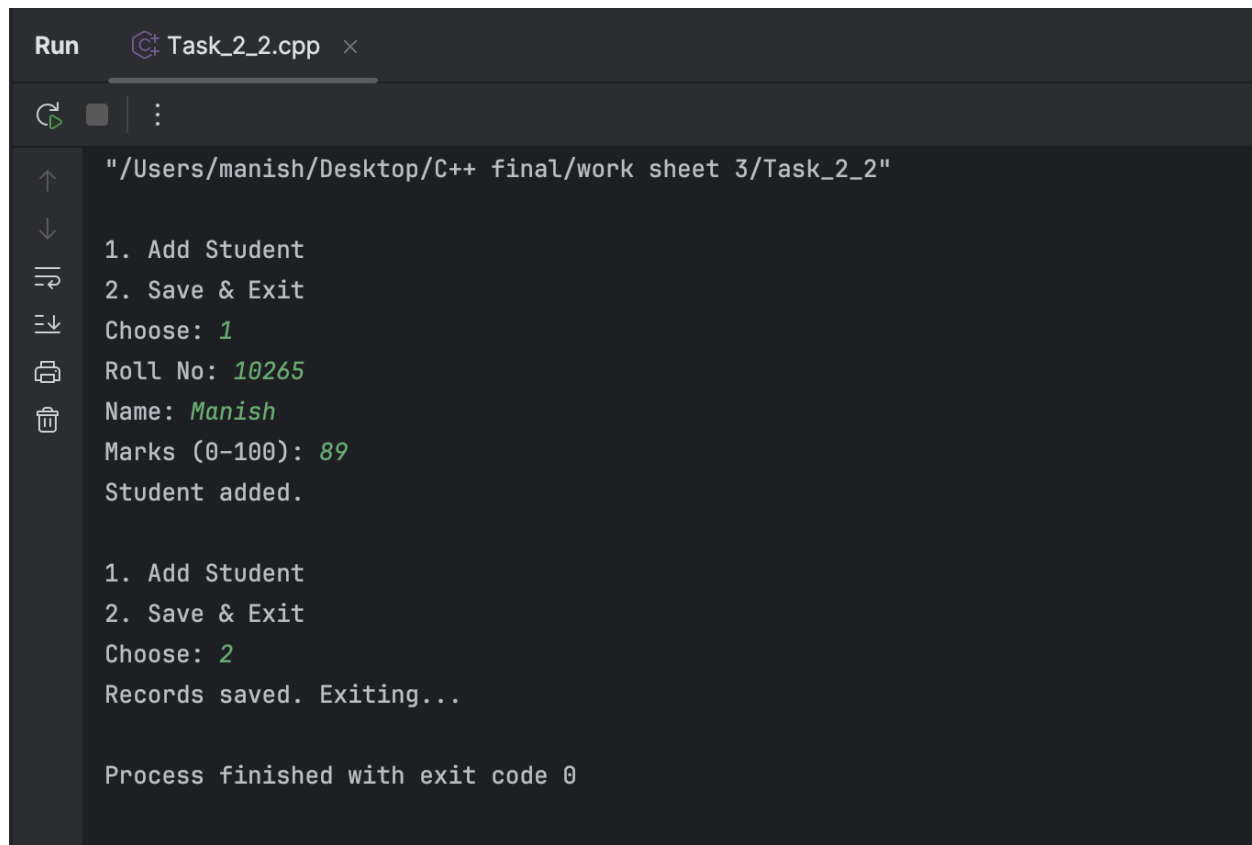
    int choice;
    do {
        cout << "\n1. Add Student\n2. Save & Exit\nChoose: ";
        cin >> choice;

        if (choice == 1) {
            Student s;
            s.input();
            students.push_back(s);
            cout << "Student added.\n";
        } else if (choice == 2) {
            Student::saveToFile(filename, students);
            cout << "Records saved. Exiting...\n";
        } else {

```

```
        cout << "Invalid choice.\n";  
    }  
  
    } while (choice != 2);  
  
    return 0;  
}
```

Output:



```
Run Task_2_2.cpp x  
"/Users/manish/Desktop/C++ final/work sheet 3/Task_2_2"  
1. Add Student  
2. Save & Exit  
Choose: 1  
Roll No: 10265  
Name: Manish  
Marks (0-100): 89  
Student added.  
  
1. Add Student  
2. Save & Exit  
Choose: 2  
Records saved. Exiting...  
  
Process finished with exit code 0
```