

**International University of Business Agriculture and Technology**

**Lab Report 9**

**Course Code: CSC 284**

**Course Name: Programming in C++ Lab**

**Submitted To:**

**Engr. A.S.M. Shakil Ahamed**

Senior Lecturer

Dept. of Computer Science and Engineering

International University of Business Agriculture and Technology

**Submitted By:**

Name: Md. Mahfujar Rahman

ID: 23303151

Section: C

**1.Code:**

#include <iostream>

using namespace std;

class Temperature

{

private:

    double celsius;

public:

    Temperature(double tempC) : celsius(tempC) {}

    double getCelsius() const { return celsius; }

    double operator-()

    {

        return (celsius \* 9.0 / 5.0) + 32;

    }

};

int main()

{

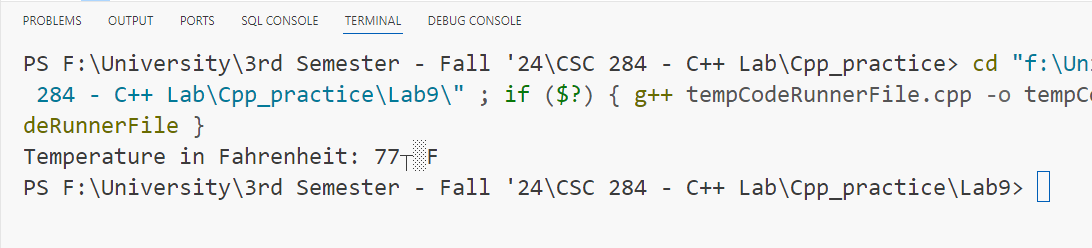
    Temperature temp(25.0);

    cout << "Temperature in Fahrenheit: " << -temp << "°F" << endl;

    return 0;

}

**Output:**

****

**2.Code:** #include <iostream>

using namespace std;

class Account

{

private:

    double balance;

public:

    Account(double initialBalance) : balance(initialBalance) {}

    double getBalance() const { return balance; }

    Account &operator++()

    {

        balance += balance \* 0.05;

        return \*this;

    }

};

int main()

{

    Account acc(1000.0);

    cout << "Initial Balance: " << acc.getBalance() << endl;

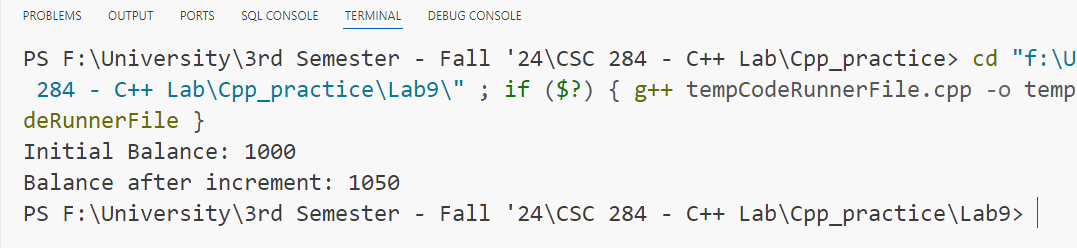
    ++acc;

    cout << "Balance after increment: " << acc.getBalance() << endl;

    return 0;

}

**Output:**



**3.Code:**

#include <iostream>

using namespace std;

class Vector

{

private:

    double x, y;

public:

    Vector(double x, double y) : x(x), y(y) {}

    void display() const

    {

        cout << "Vector: (" << x << ", " << y << ")" << endl;

    }

    Vector operator-()

    {

        return Vector(-x, -y);

    }

};

int main()

{

    Vector force(3.0, 4.0);

    force.display();

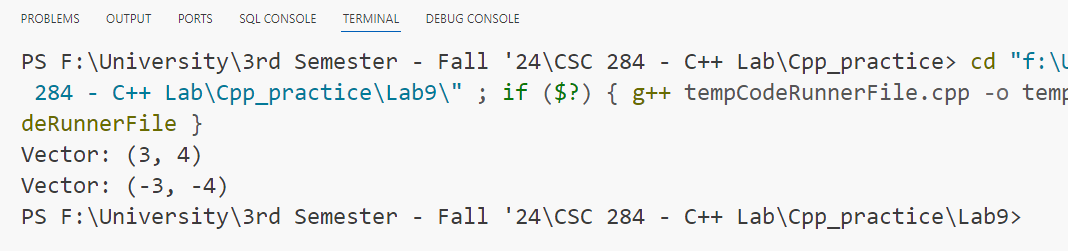
    Vector oppositeForce = -force;

    oppositeForce.display();

    return 0;

}

**Output:**

****

**4.Code:**

#include <iostream>

using namespace std;

class Item

{

private:

    int stock;

public:

    Item(int initialStock) : stock(initialStock) {}

    int getStock() const { return stock; }

    Item &operator--()

    {

        if (stock > 0)

        {

            --stock;

        }

        return \*this;

    }

};

int main()

{

    Item product(10);

    cout << "Initial stock: " << product.getStock() << endl;

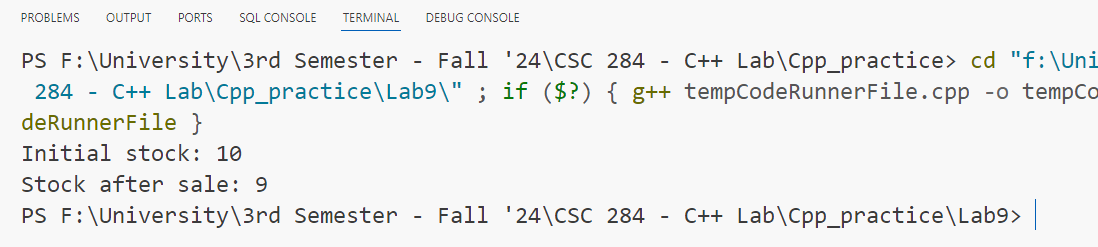
    --product;

    cout << "Stock after sale: " << product.getStock() << endl;

    return 0;

}

**Output:**

****

**5.Code:**

#include <iostream>

using namespace std;

class Signal

{

private:

    bool state;

public:

    Signal(bool initialState) : state(initialState) {}

    bool getState() const { return state; }

    Signal operator!()

    {

        return Signal(!state);

    }

};

int main()

{

    Signal signal(1);

    cout << "Initial Signal: " << signal.getState() << endl;

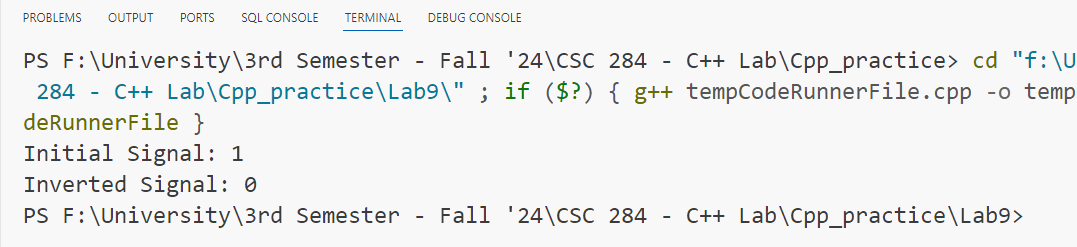
    Signal invertedSignal = !signal;

    cout << "Inverted Signal: " << invertedSignal.getState() << endl;

    return 0;

}

**Output:**

****

**6.Code:**

#include <iostream>

#include <ctime>

using namespace std;

class Date

{

private:

    int day, month, year;

public:

    Date(int d, int m, int y) : day(d), month(m), year(y) {}

    void display() const

    {

        cout << "Date: " << day << "/" << month << "/" << year << endl;

    }

    Date operator+()

    {

        struct tm date = {0};

        date.tm\_mday = day;

        date.tm\_mon = month - 1;

        date.tm\_year = year - 1900;

        mktime(&date);

        date.tm\_mday++;

        mktime(&date);

        return Date(date.tm\_mday, date.tm\_mon + 1, date.tm\_year + 1900);

    }

};

int main()

{

    Date date(7, 1, 2025);

    date.display();

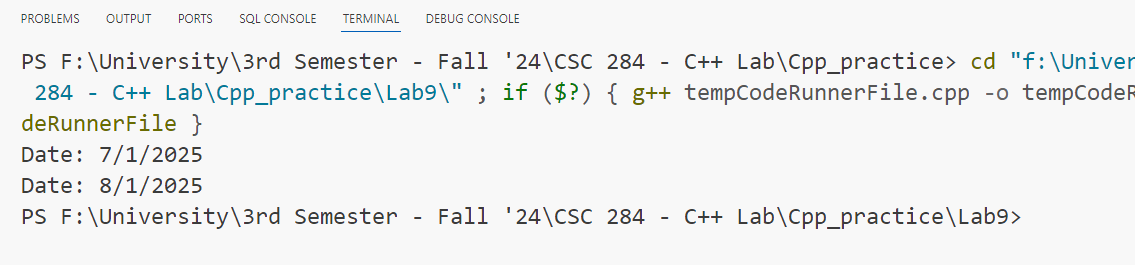
    Date nextDay = +date;

    nextDay.display();

    return 0;

}

**Output:**

****

**7.Code:**

#include <iostream>

using namespace std;

class Battery

{

private:

    int energyLevel;

public:

    Battery(int initialEnergy) : energyLevel(initialEnergy) {}

    int getEnergyLevel() const { return energyLevel; }

    Battery &operator-()

    {

        if (energyLevel > 0)

        {

            --energyLevel;

        }

        return \*this;

    }

};

int main()

{

    Battery battery(10);

    cout << "Initial Energy: " << battery.getEnergyLevel() << endl;

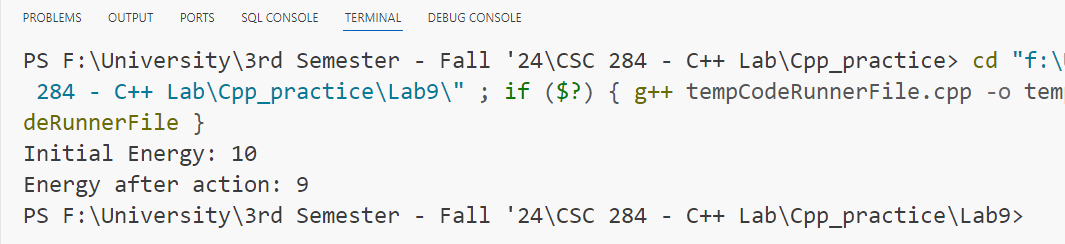
    -battery;

    cout << "Energy after action: " << battery.getEnergyLevel() << endl;

    return 0;

}

**Output:**



**8.Code:**

#include <iostream>

using namespace std;

class Car

{

private:

    int speed;

public:

    Car(int initialSpeed) : speed(initialSpeed) {}

    int getSpeed() const { return speed; }

    Car &operator--()

    {

        speed -= 10;

        if (speed < 0)

            speed = 0;

        return \*this;

    }

};

int main()

{

    Car car(50);

    cout << "Initial Speed: " << car.getSpeed() << endl;

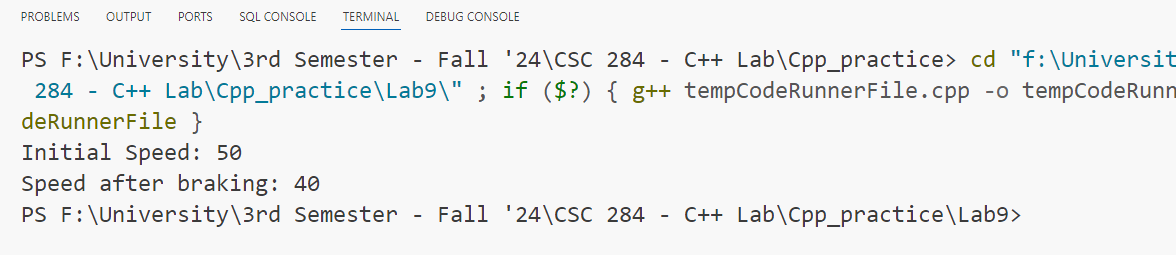
    --car;

    cout << "Speed after braking: " << car.getSpeed() << endl;

    return 0;

}

**Output:**

****

**9.Code:**

#include <iostream>

using namespace std;

class Character

{

private:

    int health;

public:

    Character(int initialHealth) : health(initialHealth) {}

    int getHealth() const { return health; }

    Character &operator+()

    {

        health += 50;

        return \*this;

    }

};

int main()

{

    Character player(100);

    cout << "Initial Health: " << player.getHealth() << endl;

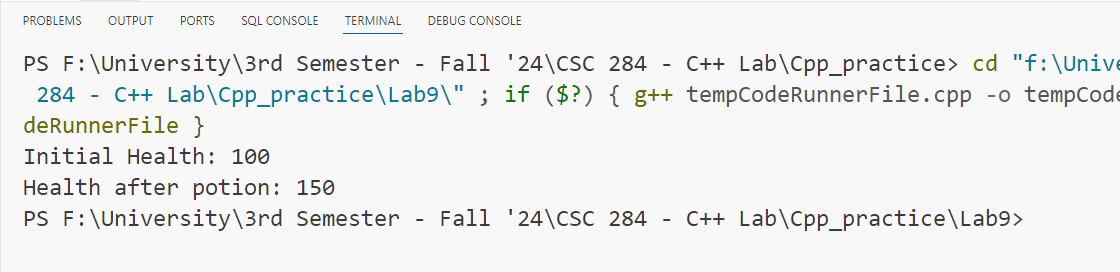
    +player;

    cout << "Health after potion: " << player.getHealth() << endl;

    return 0;

}

**Output:**

****

**10.Code:**

#include <iostream>

using namespace std;

class Task

{

private:

    bool completed;

public:

    Task(bool status) : completed(status) {}

    bool isCompleted() const { return completed; }

    Task &operator!()

    {

        completed = !completed;

        return \*this;

    }

};

int main()

{

    Task task(false);

    cout << "Task completed: " << task.isCompleted() << endl;

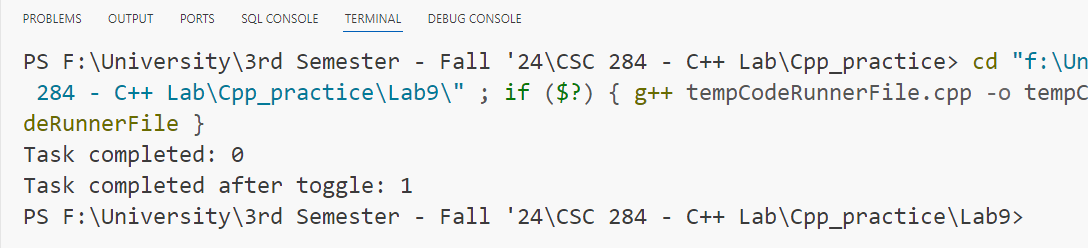
    !task;

    cout << "Task completed after toggle: " << task.isCompleted() << endl;

    return 0;

}

**Output:**

****