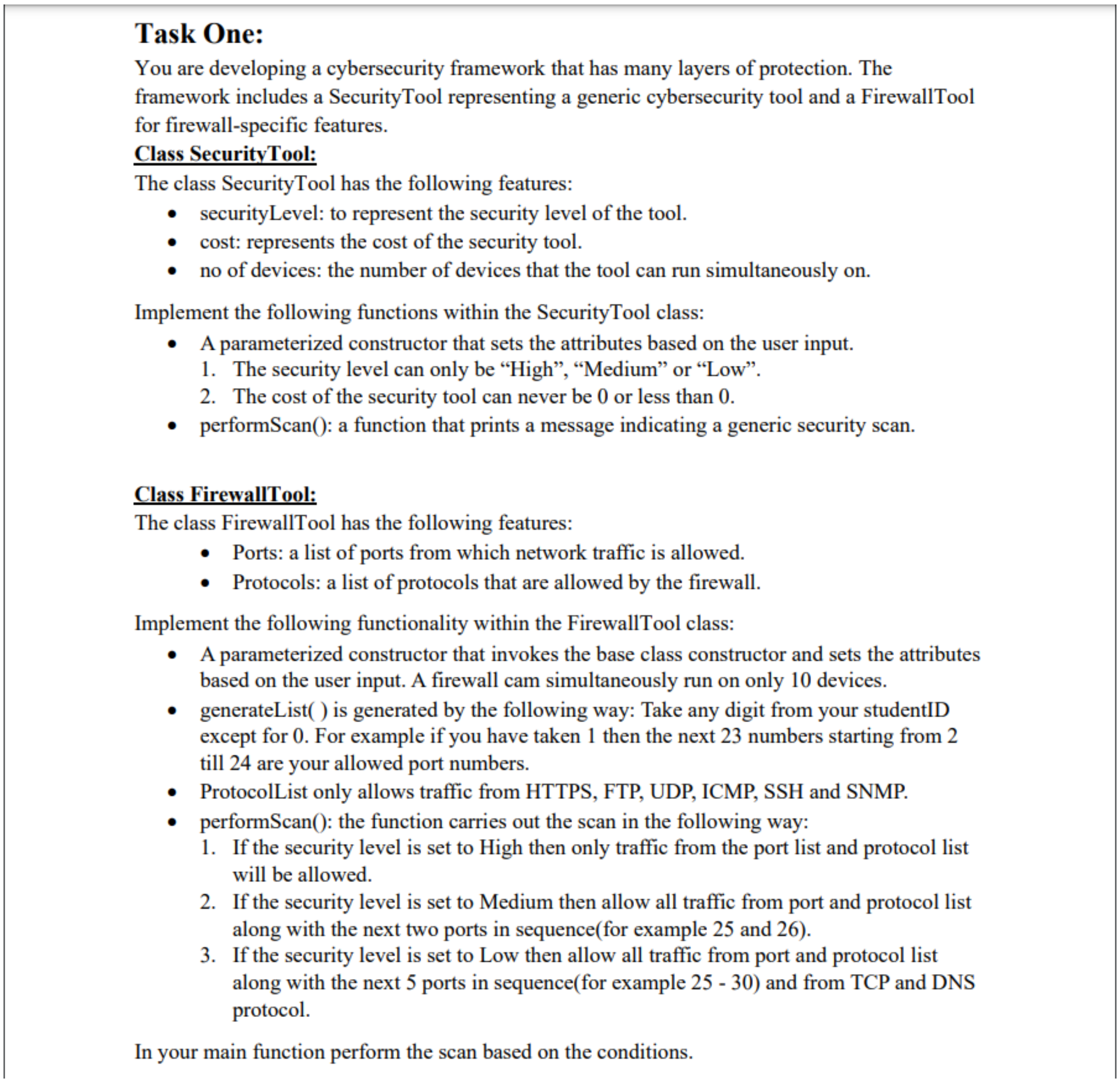
# OOP Assignment 02

## Question 01:



### Task One:

You are developing a cybersecurity framework that has many layers of protection. The framework includes a SecurityTool representing a generic cybersecurity tool and a FirewallTool for firewall-specific features.

**Class SecurityTool:**

The class SecurityTool has the following features:

- securityLevel: to represent the security level of the tool.
- cost: represents the cost of the security tool.
- no of devices: the number of devices that the tool can run simultaneously on.

Implement the following functions within the SecurityTool class:

- A parameterized constructor that sets the attributes based on the user input.
  1. The security level can only be “High”, “Medium” or “Low”.
  2. The cost of the security tool can never be 0 or less than 0.
- performScan(): a function that prints a message indicating a generic security scan.

**Class FirewallTool:**

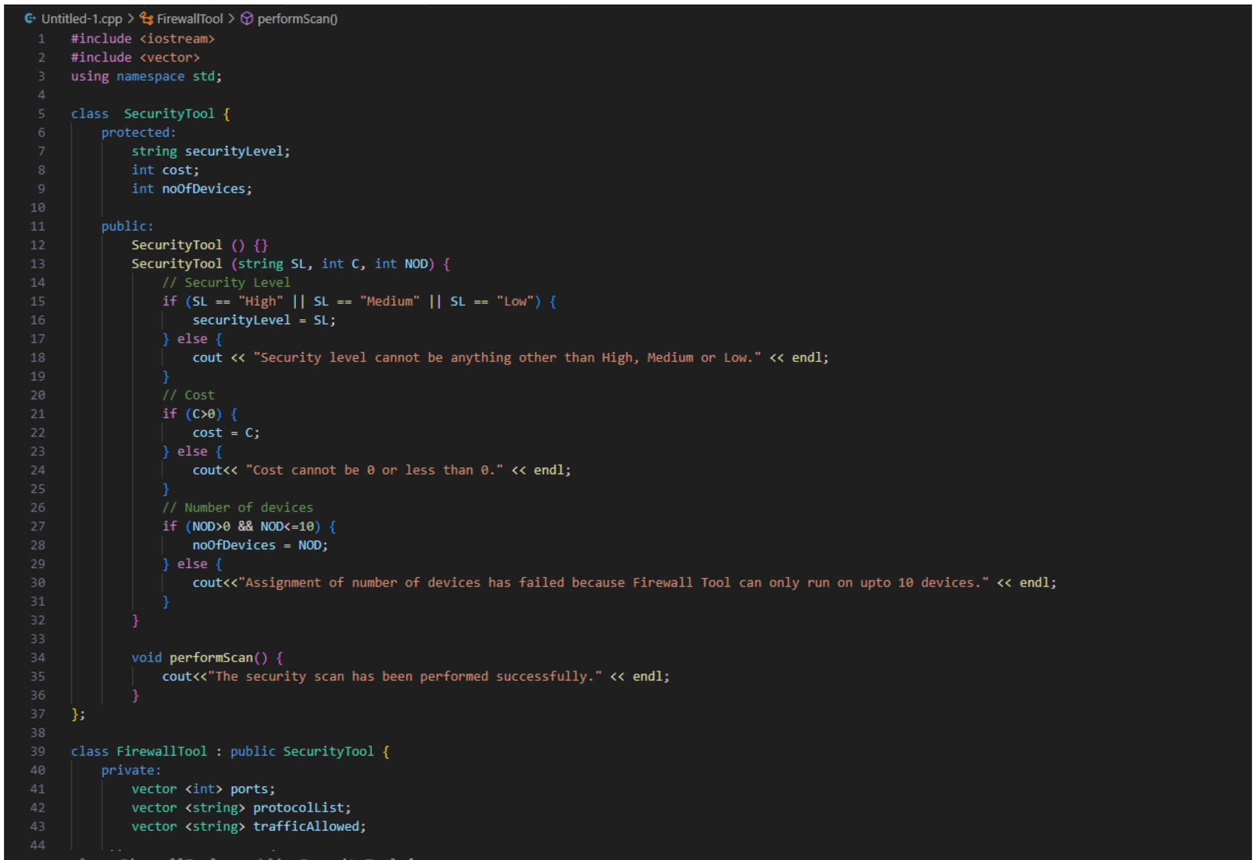
The class FirewallTool has the following features:

- Ports: a list of ports from which network traffic is allowed.
- Protocols: a list of protocols that are allowed by the firewall.

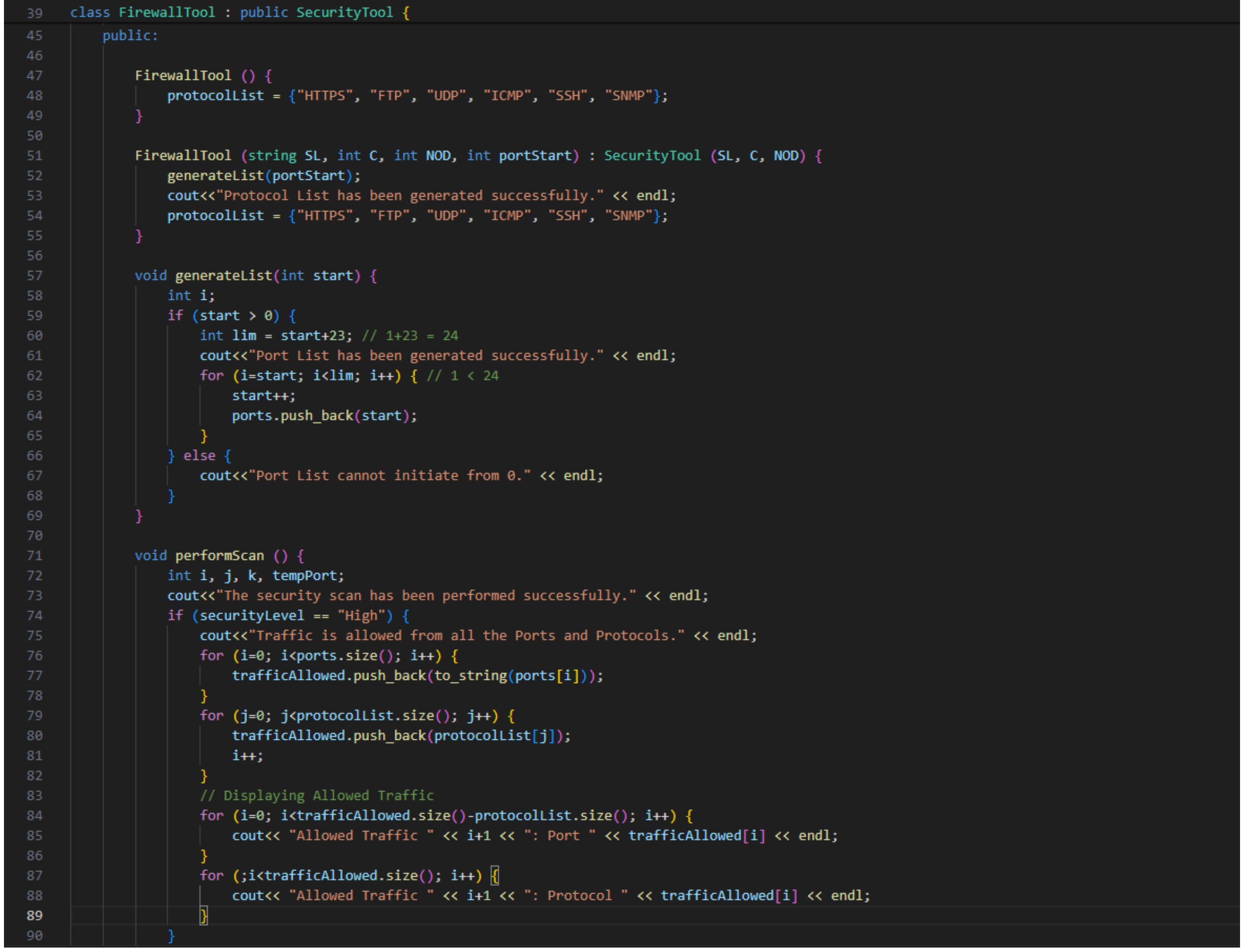
Implement the following functionality within the FirewallTool class:

- A parameterized constructor that invokes the base class constructor and sets the attributes based on the user input. A firewall cam simultaneously run on only 10 devices.
- generateList( ) is generated by the following way: Take any digit from your studentID except for 0. For example if you have taken 1 then the next 23 numbers starting from 2 till 24 are your allowed port numbers.
- ProtocolList only allows traffic from HTTPS, FTP, UDP, ICMP, SSH and SNMP.
- performScan(): the function carries out the scan in the following way:
  1. If the security level is set to High then only traffic from the port list and protocol list will be allowed.
  2. If the security level is set to Medium then allow all traffic from port and protocol list along with the next two ports in sequence(for example 25 and 26).
  3. If the security level is set to Low then allow all traffic from port and protocol list along with the next 5 ports in sequence(for example 25 - 30) and from TCP and DNS protocol.

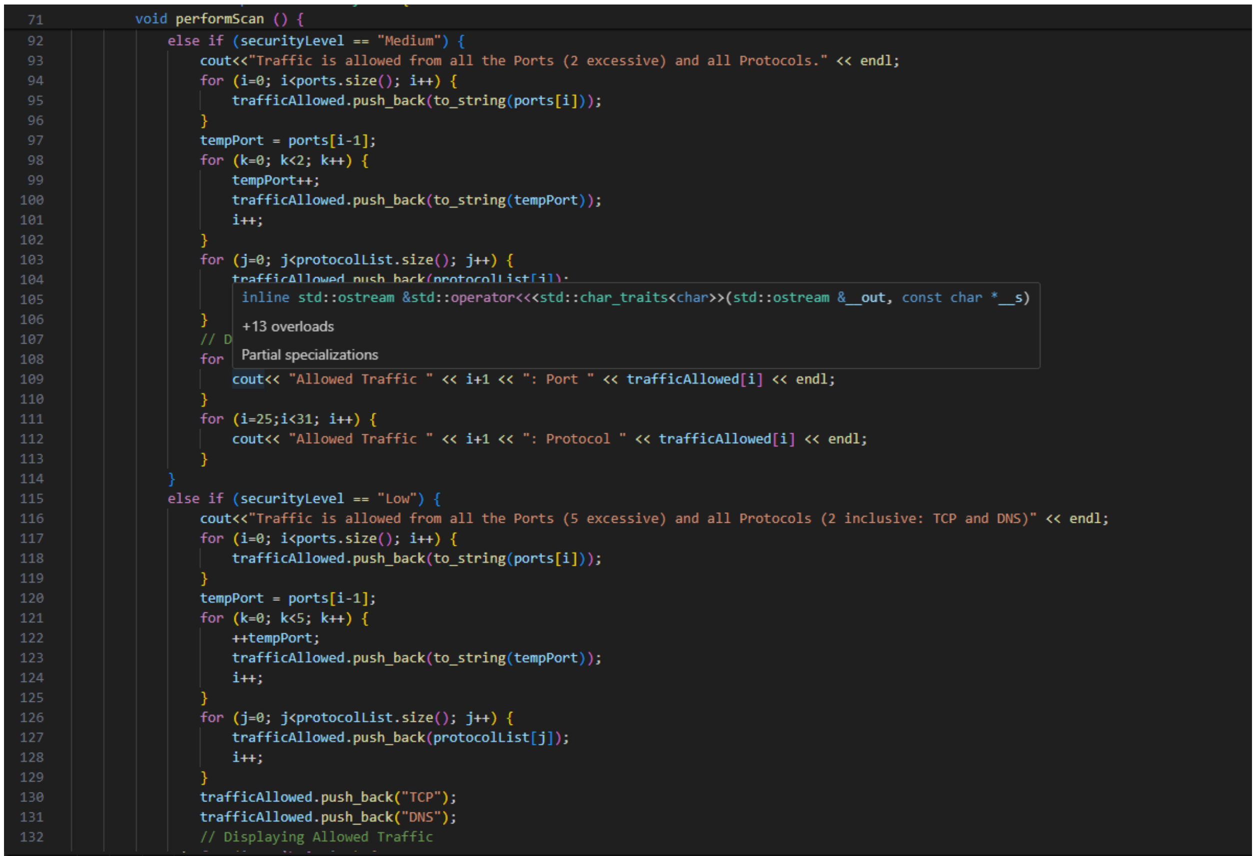
In your main function perform the scan based on the conditions.


Untitled-1.cpp > FirewallTool > performScan()

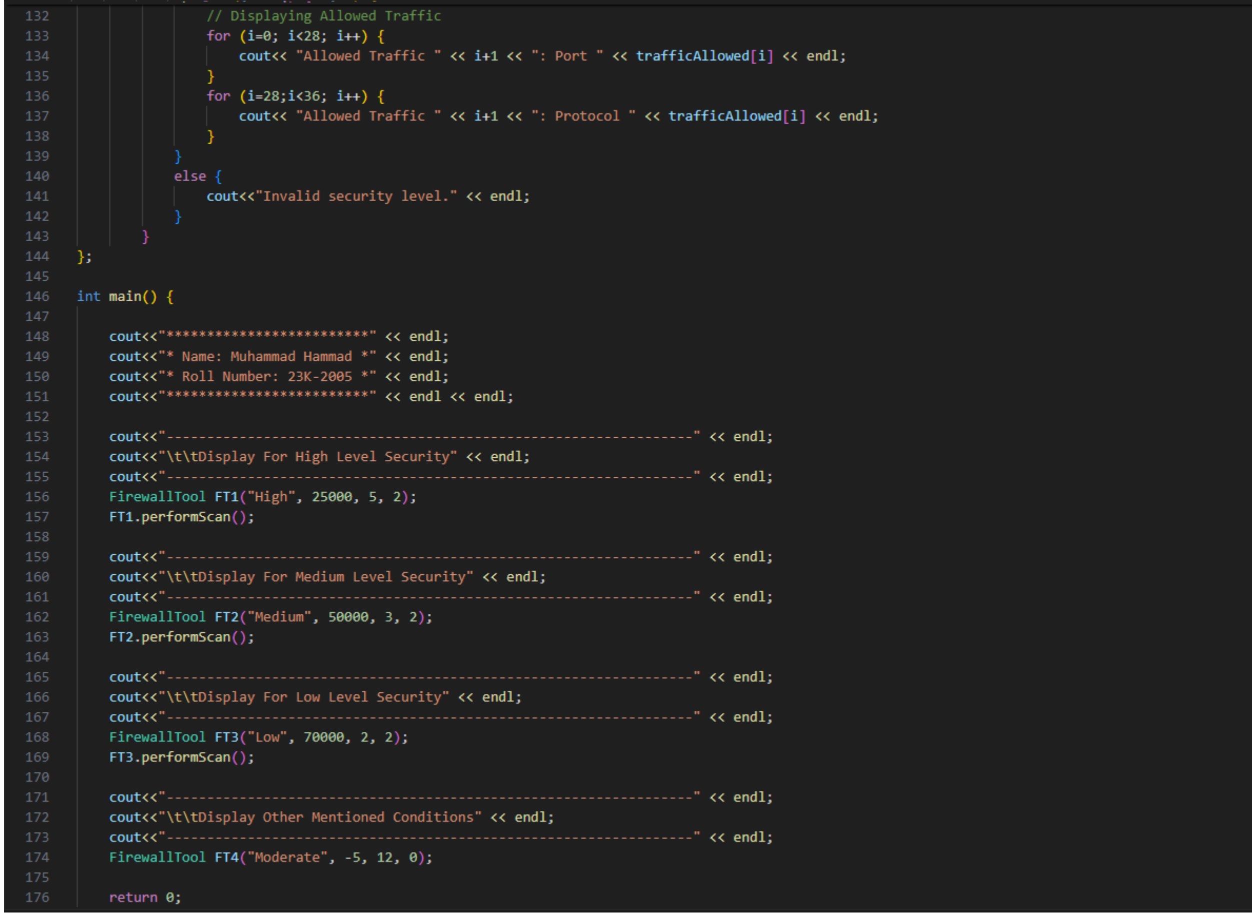
```cpp
#include <iostream>
#include <vector>
using namespace std;

class  SecurityTool {
    protected:
        string securityLevel;
        int cost;
        int noOfDevices;

    public:
        SecurityTool () {}
        SecurityTool (string SL, int C, int NOD) {
            // Security Level
            if (SL == "High" || SL == "Medium" || SL == "Low") {
                securityLevel = SL;
            } else {
                cout << "Security level cannot be anything other than High, Medium or Low." << endl;
            }
            // Cost
            if (C>0) {
                cost = C;
            } else {
                cout<< "Cost cannot be 0 or less than 0." << endl;
            }
            // Number of devices
            if (NOD>0 && NOD<=10) {
                noOfDevices = NOD;
            } else {
                cout<<"Assignment of number of devices has failed because Firewall Tool can only run on upto 10 devices." << endl;
            }
        }

        void performScan() {
            cout<<"The security scan has been performed successfully." << endl;
        }
};

class FirewallTool : public SecurityTool {
    private:
        vector <int> ports;
        vector <string> protocolList;
        vector <string> trafficAllowed;

```


class FirewallTool : public SecurityTool {

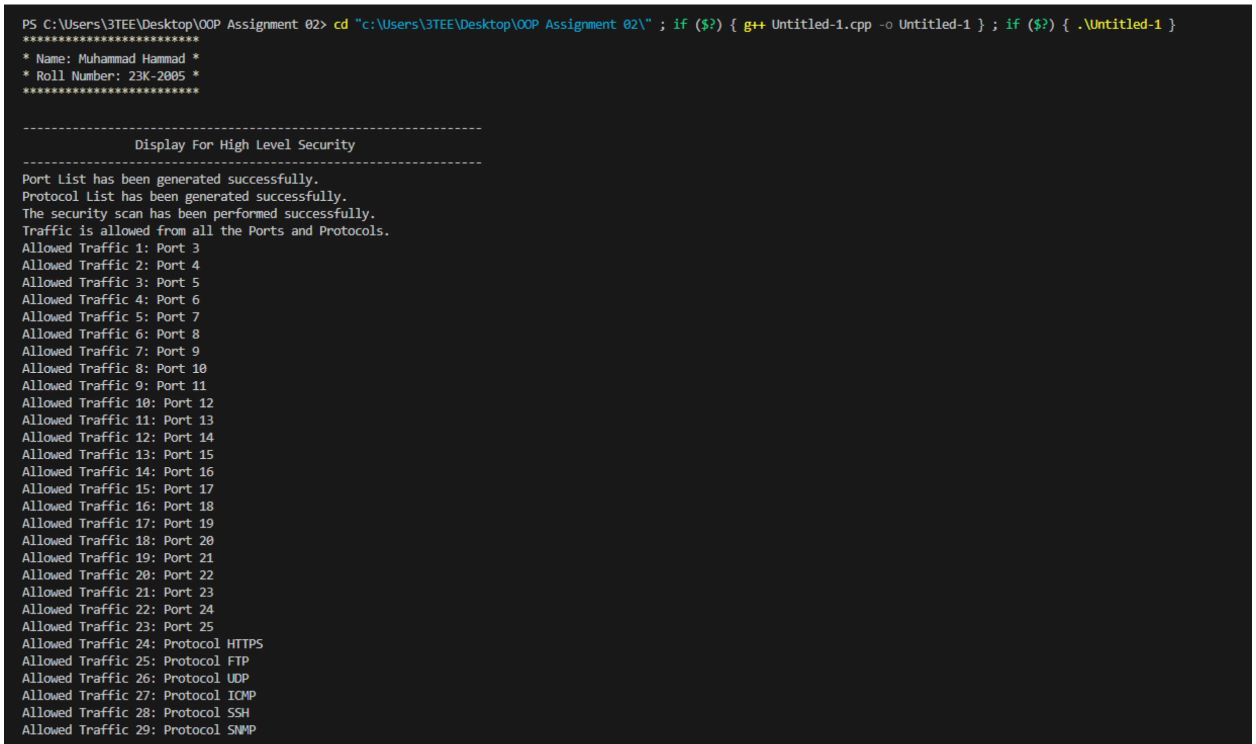
```cpp
    public:

        FirewallTool () {
            protocolList = {"HTTPS", "FTP", "UDP", "ICMP", "SSH", "SNMP"};
        }

        FirewallTool (string SL, int C, int NOD, int portStart) : SecurityTool (SL, C, NOD) {
            generateList(portStart);
            cout<<"Protocol List has been generated successfully." << endl;
            protocolList = {"HTTPS", "FTP", "UDP", "ICMP", "SSH", "SNMP"};
        }

        void generateList(int start) {
            int i;
            if (start > 0) {
                int lim = start+23; // 1+23 = 24
                cout<<"Port List has been generated successfully." << endl;
                for (i=start; i<lim; i++) { // 1 < 24
                    start++;
                    ports.push_back(start);
                }
            } else {
                cout<<"Port List cannot initiate from 0." << endl;
            }
        }

        void performScan () {
            int i, j, k, tempPort;
            cout<<"The security scan has been performed successfully." << endl;
            if (securityLevel == "High") {
                cout<<"Traffic is allowed from all the Ports and Protocols." << endl;
                for (i=0; i<ports.size(); i++) {
                    trafficAllowed.push_back(to_string(ports[i]));
                }
                for (j=0; j<protocolList.size(); j++) {
                    trafficAllowed.push_back(protocolList[j]);
                    i++;
                }
                // Displaying Allowed Traffic
                for (i=0; i<trafficAllowed.size()-protocolList.size(); i++) {
                    cout<< "Allowed Traffic " << i+1 << ": Port " << trafficAllowed[i] << endl;
                }
                for (;i<trafficAllowed.size(); i++) {
                    cout<< "Allowed Traffic " << i+1 << ": Protocol " << trafficAllowed[i] << endl;
                }
            }
```



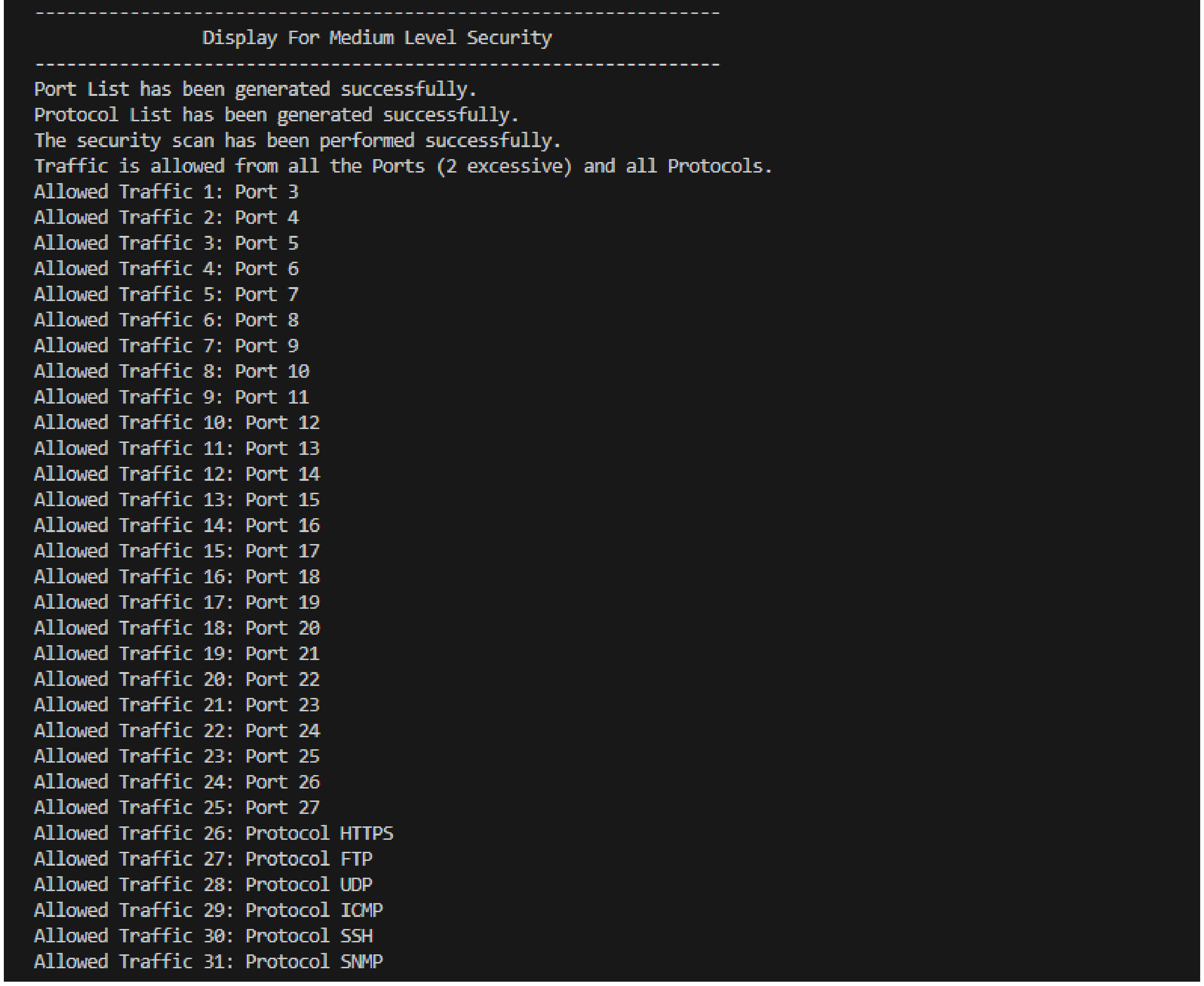
```
        void performScan () {
            else if (securityLevel == "Medium") {
                cout<<"Traffic is allowed from all the Ports (2 excessive) and all Protocols." << endl;
                for (i=0; i<ports.size(); i++) {
                    trafficAllowed.push_back(to_string(ports[i]));
                }
                tempPort = ports[i-1];
                for (k=0; k<2; k++) {
                    tempPort++;
                    trafficAllowed.push_back(to_string(tempPort));
                    i++;
                }
                for (j=0; j<protocolList.size(); j++) {
                    trafficAllowed.push_back(protocolList[j]);
                  inline std::ostream &std::operator<<<std::char_traits<char>>(std::ostream &__out, const char *__s)
                }
                  +13 overloads
                // D
                for   Partial specializations
                    cout<< "Allowed Traffic " << i+1 << ": Port " << trafficAllowed[i] << endl;
                }
                for (i=25;i<31; i++) {
                    cout<< "Allowed Traffic " << i+1 << ": Protocol " << trafficAllowed[i] << endl;
                }
            }
            else if (securityLevel == "Low") {
                cout<<"Traffic is allowed from all the Ports (5 excessive) and all Protocols (2 inclusive: TCP and DNS)" << endl;
                for (i=0; i<ports.size(); i++) {
                    trafficAllowed.push_back(to_string(ports[i]));
                }
                tempPort = ports[i-1];
                for (k=0; k<5; k++) {
                    ++tempPort;
                    trafficAllowed.push_back(to_string(tempPort));
                    i++;
                }
                for (j=0; j<protocolList.size(); j++) {
                    trafficAllowed.push_back(protocolList[j]);
                    i++;
                }
                trafficAllowed.push_back("TCP");
                trafficAllowed.push_back("DNS");
                // Displaying Allowed Traffic
```



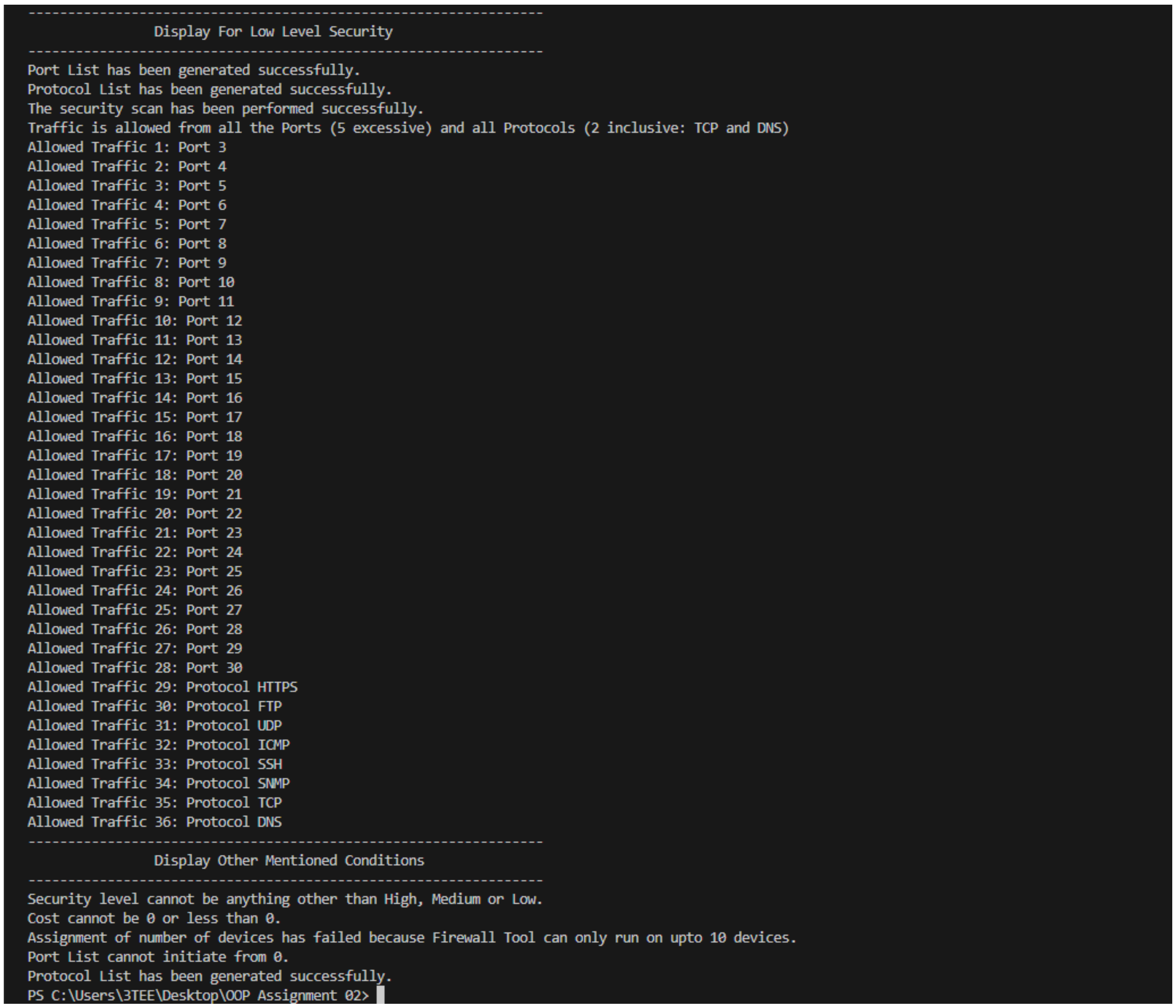
```
                // Displaying Allowed Traffic
                for (i=0; i<28; i++) {
                    cout<< "Allowed Traffic " << i+1 << ": Port " << trafficAllowed[i] << endl;
                }
                for (i=28;i<36; i++) {
                    cout<< "Allowed Traffic " << i+1 << ": Protocol " << trafficAllowed[i] << endl;
                }
            }
            else {
                cout<<"Invalid security level." << endl;
            }
        }
};

int main() {

    cout<<"************************" << endl;
    cout<<"* Name: Muhammad Hammad *" << endl;
    cout<<"* Roll Number: 23K-2005 *" << endl;
    cout<<"************************" << endl << endl;

    cout<<"-----------------------------------------------------------------" << endl;
    cout<<"\t\tDisplay For High Level Security" << endl;
    cout<<"-----------------------------------------------------------------" << endl;
    FirewallTool FT1("High", 25000, 5, 2);
    FT1.performScan();

    cout<<"-----------------------------------------------------------------" << endl;
    cout<<"\t\tDisplay For Medium Level Security" << endl;
    cout<<"-----------------------------------------------------------------" << endl;
    FirewallTool FT2("Medium", 50000, 3, 2);
    FT2.performScan();

    cout<<"-----------------------------------------------------------------" << endl;
    cout<<"\t\tDisplay For Low Level Security" << endl;
    cout<<"-----------------------------------------------------------------" << endl;
    FirewallTool FT3("Low", 70000, 2, 2);
    FT3.performScan();

    cout<<"-----------------------------------------------------------------" << endl;
    cout<<"\t\tDisplay Other Mentioned Conditions" << endl;
    cout<<"-----------------------------------------------------------------" << endl;
    FirewallTool FT4("Moderate", -5, 12, 0);

    return 0;
```



```
PS C:\Users\3TEE\Desktop\OOP Assignment 02> cd "c:\Users\3TEE\Desktop\OOP Assignment 02\" ; if ($?) { g++ Untitled-1.cpp -o Untitled-1 } ; if ($?) { .\Untitled-1 }
*************************
* Name: Muhammad Hammad *
* Roll Number: 23K-2005 *
*************************

-----------------------------------------------------------------
               Display For High Level Security
-----------------------------------------------------------------
Port List has been generated successfully.
Protocol List has been generated successfully.
The security scan has been performed successfully.
Traffic is allowed from all the Ports and Protocols.
Allowed Traffic 1: Port 3
Allowed Traffic 2: Port 4
Allowed Traffic 3: Port 5
Allowed Traffic 4: Port 6
Allowed Traffic 5: Port 7
Allowed Traffic 6: Port 8
Allowed Traffic 7: Port 9
Allowed Traffic 8: Port 10
Allowed Traffic 9: Port 11
Allowed Traffic 10: Port 12
Allowed Traffic 11: Port 13
Allowed Traffic 12: Port 14
Allowed Traffic 13: Port 15
Allowed Traffic 14: Port 16
Allowed Traffic 15: Port 17
Allowed Traffic 16: Port 18
Allowed Traffic 17: Port 19
Allowed Traffic 18: Port 20
Allowed Traffic 19: Port 21
Allowed Traffic 20: Port 22
Allowed Traffic 21: Port 23
Allowed Traffic 22: Port 24
Allowed Traffic 23: Port 25
Allowed Traffic 24: Protocol HTTPS
Allowed Traffic 25: Protocol FTP
Allowed Traffic 26: Protocol UDP
Allowed Traffic 27: Protocol ICMP
Allowed Traffic 28: Protocol SSH
Allowed Traffic 29: Protocol SNMP
```

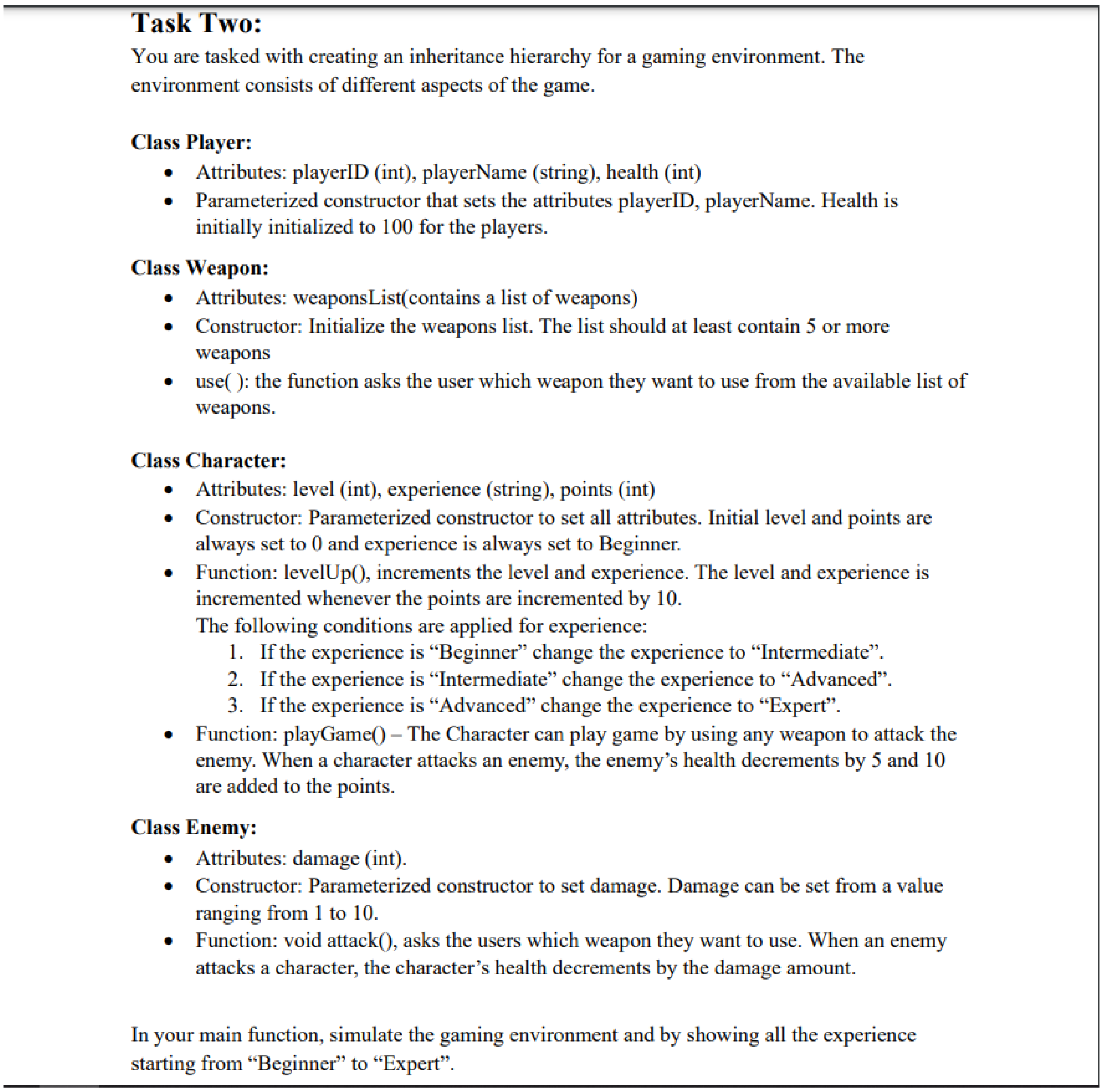


```
-----------------------------------------------------------------
               Display For Medium Level Security
-----------------------------------------------------------------
Port List has been generated successfully.
Protocol List has been generated successfully.
The security scan has been performed successfully.
Traffic is allowed from all the Ports (2 excessive) and all Protocols.
Allowed Traffic 1: Port 3
Allowed Traffic 2: Port 4
Allowed Traffic 3: Port 5
Allowed Traffic 4: Port 6
Allowed Traffic 5: Port 7
Allowed Traffic 6: Port 8
Allowed Traffic 7: Port 9
Allowed Traffic 8: Port 10
Allowed Traffic 9: Port 11
Allowed Traffic 10: Port 12
Allowed Traffic 11: Port 13
Allowed Traffic 12: Port 14
Allowed Traffic 13: Port 15
Allowed Traffic 14: Port 16
Allowed Traffic 15: Port 17
Allowed Traffic 16: Port 18
Allowed Traffic 17: Port 19
Allowed Traffic 18: Port 20
Allowed Traffic 19: Port 21
Allowed Traffic 20: Port 22
Allowed Traffic 21: Port 23
Allowed Traffic 22: Port 24
Allowed Traffic 23: Port 25
Allowed Traffic 24: Port 26
Allowed Traffic 25: Port 27
Allowed Traffic 26: Protocol HTTPS
Allowed Traffic 27: Protocol FTP
Allowed Traffic 28: Protocol UDP
Allowed Traffic 29: Protocol ICMP
Allowed Traffic 30: Protocol SSH
Allowed Traffic 31: Protocol SNMP
```



```
-----------------------------------------------------------------
                Display For Low Level Security
-----------------------------------------------------------------
Port List has been generated successfully.
Protocol List has been generated successfully.
The security scan has been performed successfully.
Traffic is allowed from all the Ports (5 excessive) and all Protocols (2 inclusive: TCP and DNS)
Allowed Traffic 1: Port 3
Allowed Traffic 2: Port 4
Allowed Traffic 3: Port 5
Allowed Traffic 4: Port 6
Allowed Traffic 5: Port 7
Allowed Traffic 6: Port 8
Allowed Traffic 7: Port 9
Allowed Traffic 8: Port 10
Allowed Traffic 9: Port 11
Allowed Traffic 10: Port 12
Allowed Traffic 11: Port 13
Allowed Traffic 12: Port 14
Allowed Traffic 13: Port 15
Allowed Traffic 14: Port 16
Allowed Traffic 15: Port 17
Allowed Traffic 16: Port 18
Allowed Traffic 17: Port 19
Allowed Traffic 18: Port 20
Allowed Traffic 19: Port 21
Allowed Traffic 20: Port 22
Allowed Traffic 21: Port 23
Allowed Traffic 22: Port 24
Allowed Traffic 23: Port 25
Allowed Traffic 24: Port 26
Allowed Traffic 25: Port 27
Allowed Traffic 26: Port 28
Allowed Traffic 27: Port 29
Allowed Traffic 28: Port 30
Allowed Traffic 29: Protocol HTTPS
Allowed Traffic 30: Protocol FTP
Allowed Traffic 31: Protocol UDP
Allowed Traffic 32: Protocol ICMP
Allowed Traffic 33: Protocol SSH
Allowed Traffic 34: Protocol SNMP
Allowed Traffic 35: Protocol TCP
Allowed Traffic 36: Protocol DNS
-----------------------------------------------------------------
                Display Other Mentioned Conditions
-----------------------------------------------------------------
Security level cannot be anything other than High, Medium or Low.
Cost cannot be 0 or less than 0.
Assignment of number of devices has failed because Firewall Tool can only run on upto 10 devices.
Port List cannot initiate from 0.
Protocol List has been generated successfully.
PS C:\Users\3TEE\Desktop\OOP Assignment 02>
```

## Question 02:



### Task Two:

You are tasked with creating an inheritance hierarchy for a gaming environment. The environment consists of different aspects of the game.

**Class Player:**

- Attributes: playerID (int), playerName (string), health (int)
- Parameterized constructor that sets the attributes playerID, playerName. Health is initially initialized to 100 for the players.

**Class Weapon:**

- Attributes: weaponsList(contains a list of weapons)
- Constructor: Initialize the weapons list. The list should at least contain 5 or more weapons
- use( ): the function asks the user which weapon they want to use from the available list of weapons.

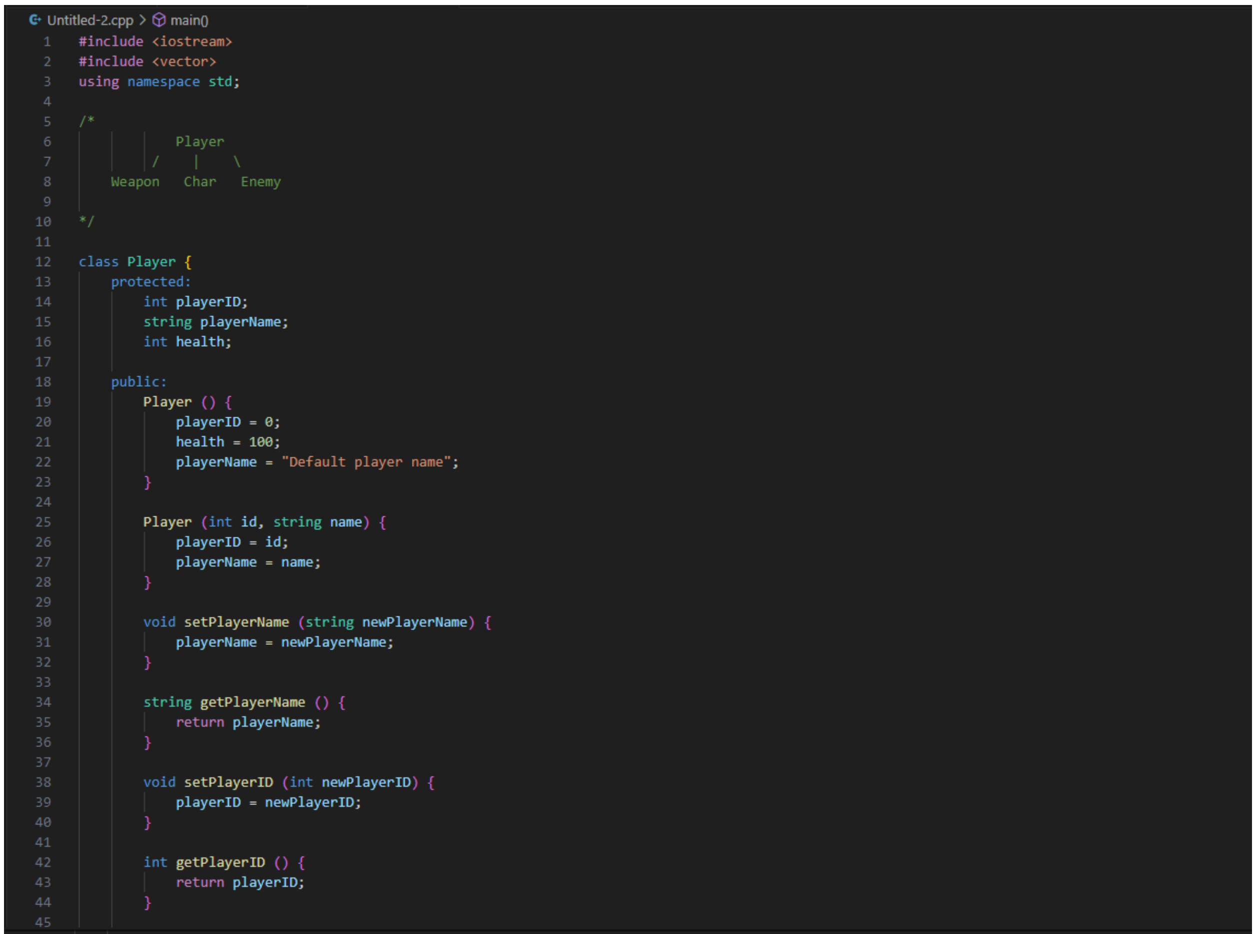
**Class Character:**

- Attributes: level (int), experience (string), points (int)
- Constructor: Parameterized constructor to set all attributes. Initial level and points are always set to 0 and experience is always set to Beginner.
- Function: levelUp(), increments the level and experience. The level and experience is incremented whenever the points are incremented by 10.
  The following conditions are applied for experience:
  1. If the experience is "Beginner" change the experience to "Intermediate".
  2. If the experience is "Intermediate" change the experience to "Advanced".
  3. If the experience is "Advanced" change the experience to "Expert".
- Function: playGame() – The Character can play game by using any weapon to attack the enemy. When a character attacks an enemy, the enemy's health decrements by 5 and 10 are added to the points.

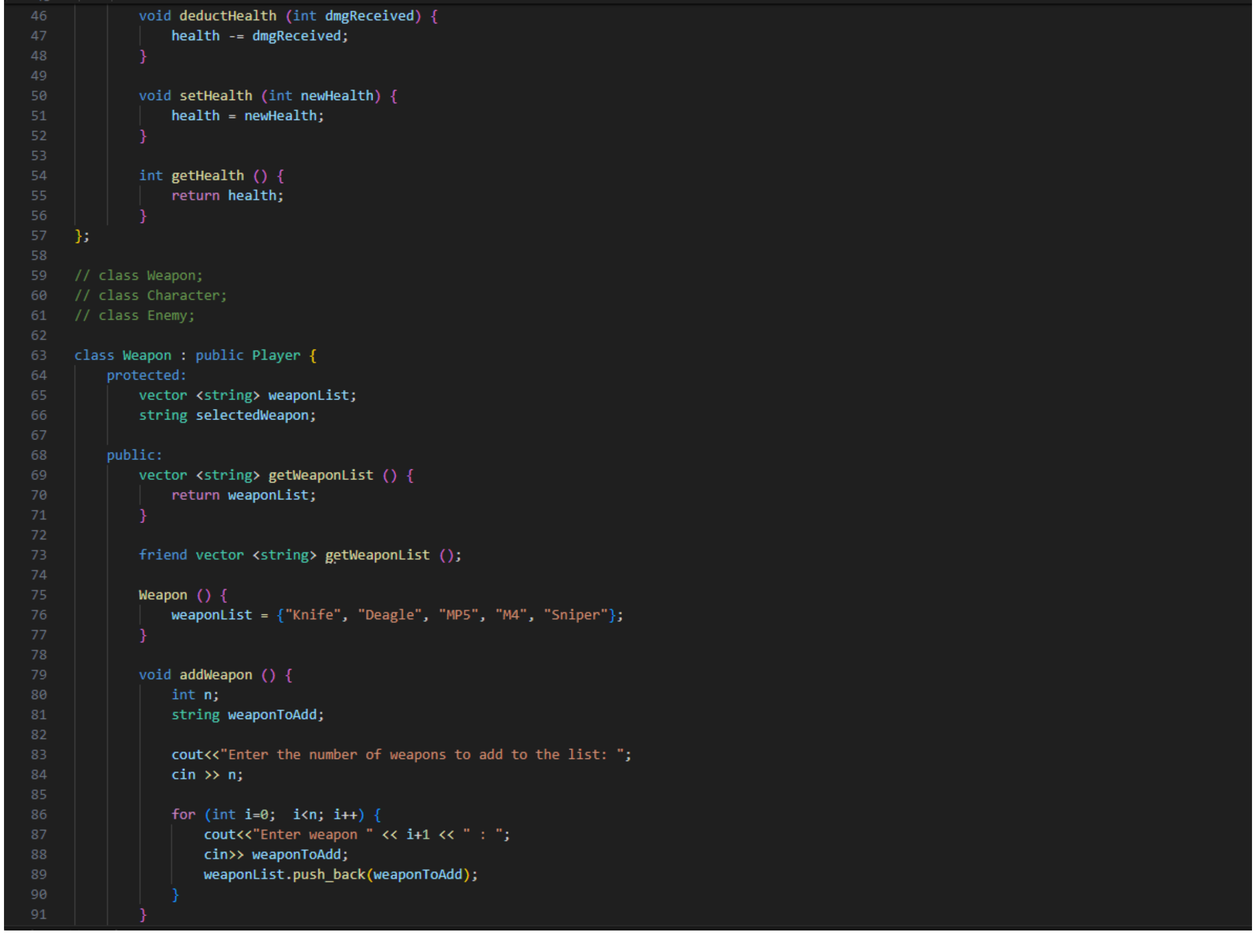
**Class Enemy:**

- Attributes: damage (int).
- Constructor: Parameterized constructor to set damage. Damage can be set from a value ranging from 1 to 10.
- Function: void attack(), asks the users which weapon they want to use. When an enemy attacks a character, the character's health decrements by the damage amount.

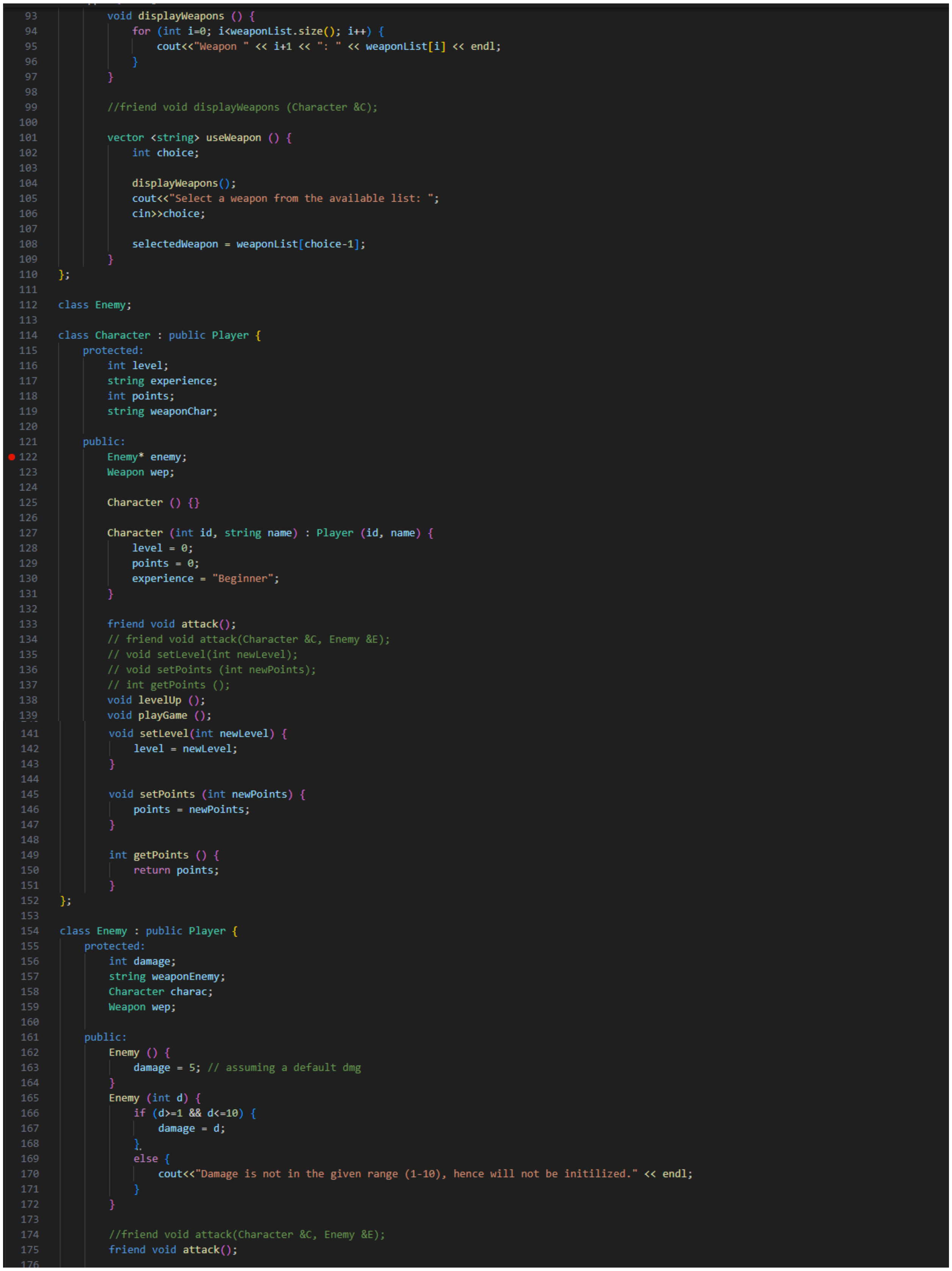
In your main function, simulate the gaming environment and by showing all the experience starting from "Beginner" to "Expert".



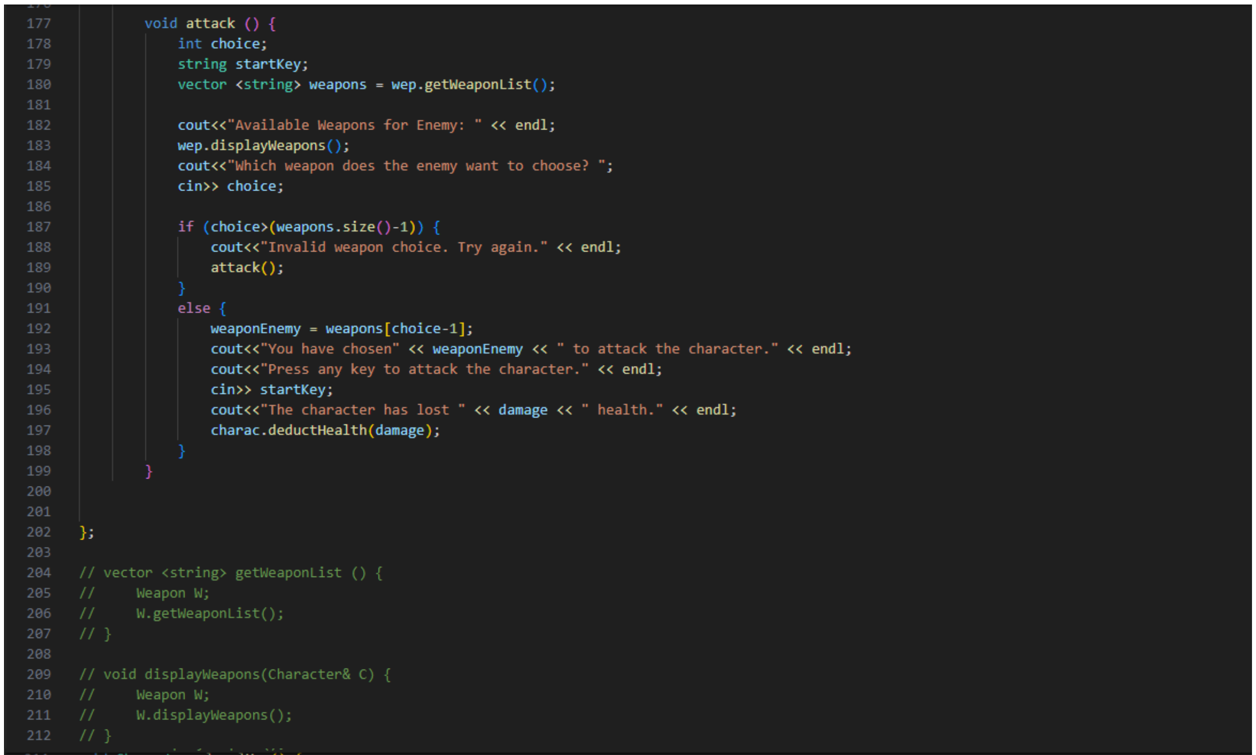
```cpp
#include <iostream>
#include <vector>
using namespace std;

/*
        Player
    /     |     \
Weapon   Char   Enemy

*/

class Player {
    protected:
        int playerID;
        string playerName;
        int health;

    public:
        Player () {
            playerID = 0;
            health = 100;
            playerName = "Default player name";
        }

        Player (int id, string name) {
            playerID = id;
            playerName = name;
        }

        void setPlayerName (string newPlayerName) {
            playerName = newPlayerName;
        }

        string getPlayerName () {
            return playerName;
        }

        void setPlayerID (int newPlayerID) {
            playerID = newPlayerID;
        }

        int getPlayerID () {
            return playerID;
        }

```



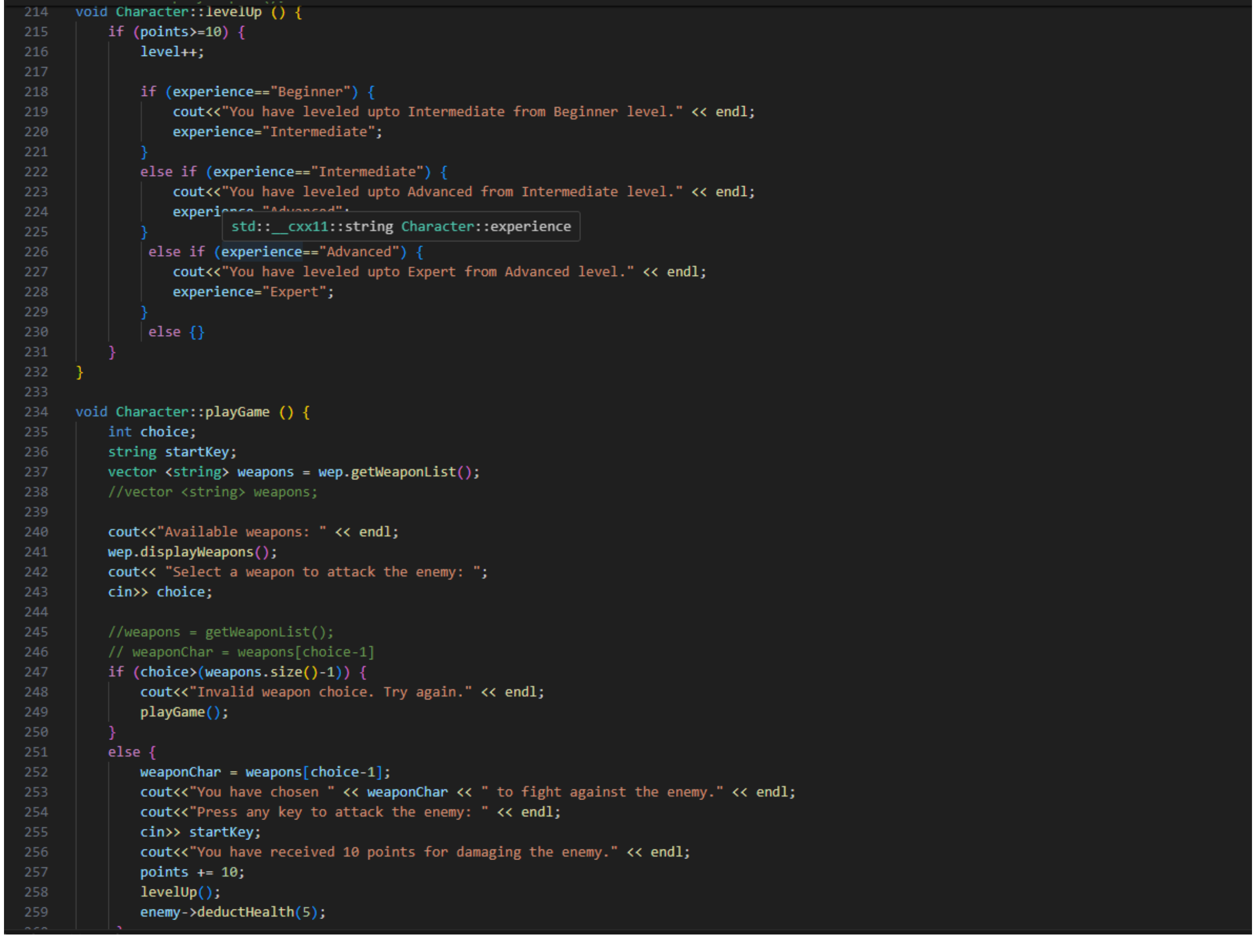
```cpp
        void deductHealth (int dmgReceived) {
            health -= dmgReceived;
        }

        void setHealth (int newHealth) {
            health = newHealth;
        }

        int getHealth () {
            return health;
        }
};

// class Weapon;
// class Character;
// class Enemy;

class Weapon : public Player {
    protected:
        vector <string> weaponList;
        string selectedWeapon;

    public:
        vector <string> getWeaponList () {
            return weaponList;
        }

        friend vector <string> getWeaponList ();

        Weapon () {
            weaponList = {"Knife", "Deagle", "MP5", "M4", "Sniper"};
        }

        void addWeapon () {
            int n;
            string weaponToAdd;

            cout<<"Enter the number of weapons to add to the list: ";
            cin >> n;

            for (int i=0;  i<n; i++) {
                cout<<"Enter weapon " << i+1 << " : ";
                cin>> weaponToAdd;
                weaponList.push_back(weaponToAdd);
            }
        }
```



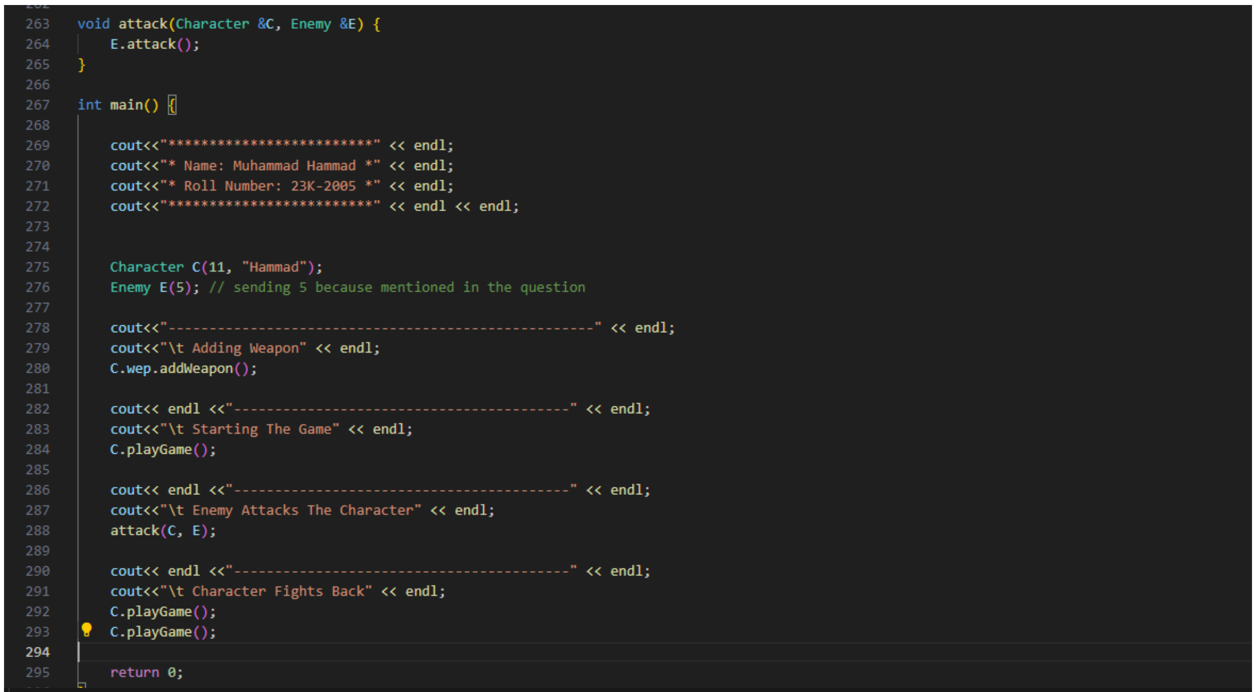
```cpp
    void displayWeapons () {
        for (int i=0; i<weaponList.size(); i++) {
            cout<<"Weapon " << i+1 << ": " << weaponList[i] << endl;
        }
    }

    //friend void displayWeapons (Character &C);

    vector <string> useWeapon () {
        int choice;

        displayWeapons();
        cout<<"Select a weapon from the available list: ";
        cin>>choice;

        selectedWeapon = weaponList[choice-1];
    }
};

class Enemy;

class Character : public Player {
    protected:
        int level;
        string experience;
        int points;
        string weaponChar;

    public:
        Enemy* enemy;
        Weapon wep;

        Character () {}

        Character (int id, string name) : Player (id, name) {
            level = 0;
            points = 0;
            experience = "Beginner";
        }

        friend void attack();
        // friend void attack(Character &C, Enemy &E);
        // void setLevel(int newLevel);
        // void setPoints (int newPoints);
        // int getPoints ();
        void levelUp ();
        void playGame ();
        void setLevel(int newLevel) {
            level = newLevel;
        }

        void setPoints (int newPoints) {
            points = newPoints;
        }

        int getPoints () {
            return points;
        }
};

class Enemy : public Player {
    protected:
        int damage;
        string weaponEnemy;
        Character charac;
        Weapon wep;

    public:
        Enemy () {
            damage = 5; // assuming a default dmg
        }
        Enemy (int d) {
            if (d>=1 && d<=10) {
                damage = d;
            }
            else {
                cout<<"Damage is not in the given range (1-10), hence will not be initilized." << endl;
            }
        }

        //friend void attack(Character &C, Enemy &E);
        friend void attack();
```



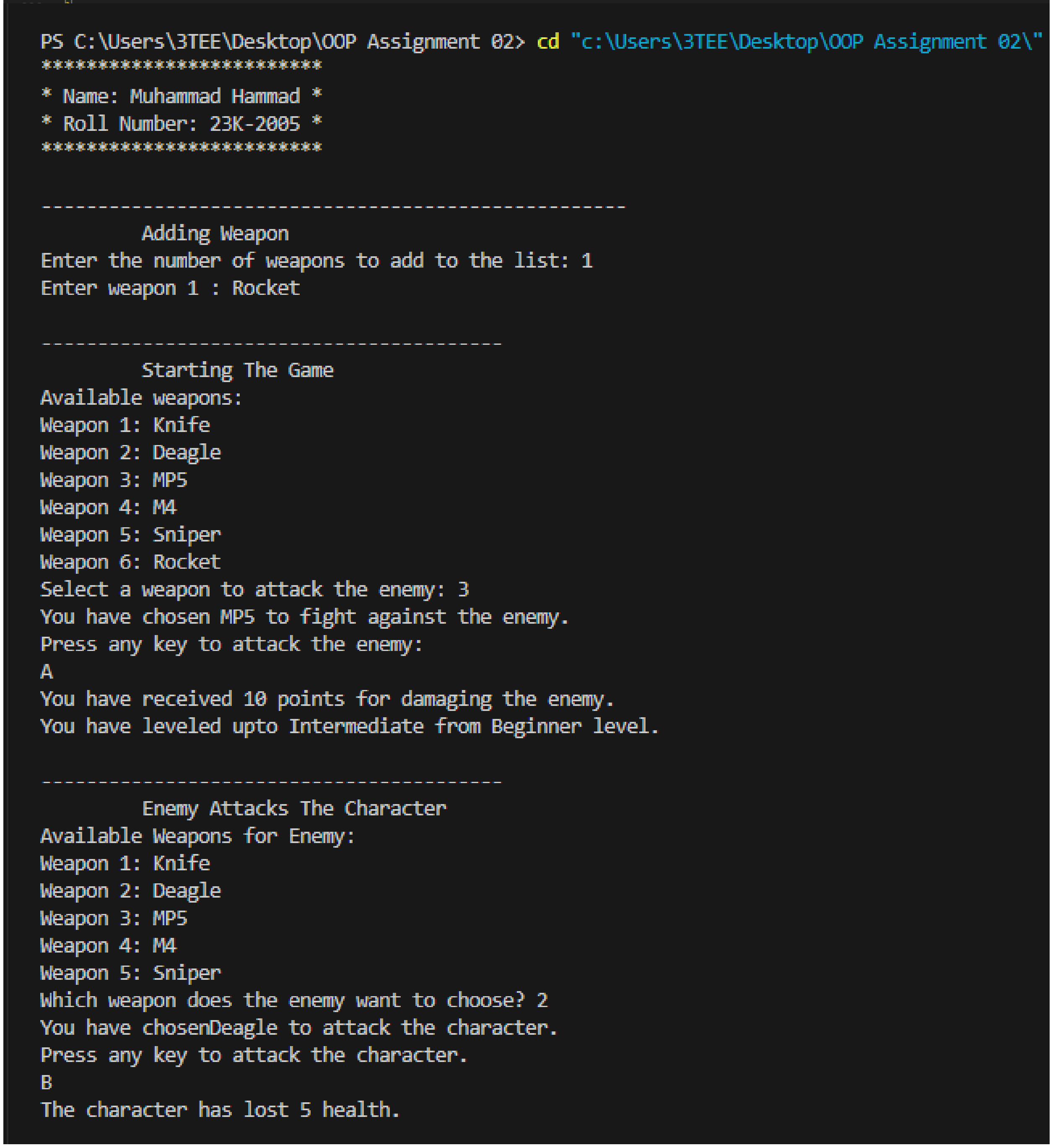
```cpp
        void attack () {
            int choice;
            string startKey;
            vector <string> weapons = wep.getWeaponList();

            cout<<"Available Weapons for Enemy: " << endl;
            wep.displayWeapons();
            cout<<"Which weapon does the enemy want to choose? ";
            cin>> choice;

            if (choice>(weapons.size()-1)) {
                cout<<"Invalid weapon choice. Try again." << endl;
                attack();
            }
            else {
                weaponEnemy = weapons[choice-1];
                cout<<"You have chosen" << weaponEnemy << " to attack the character." << endl;
                cout<<"Press any key to attack the character." << endl;
                cin>> startKey;
                cout<<"The character has lost " << damage << " health." << endl;
                charac.deductHealth(damage);
            }
        }


};

// vector <string> getWeaponList () {
//     Weapon W;
//     W.getWeaponList();
// }

// void displayWeapons(Character& C) {
//     Weapon W;
//     W.displayWeapons();
// }
```



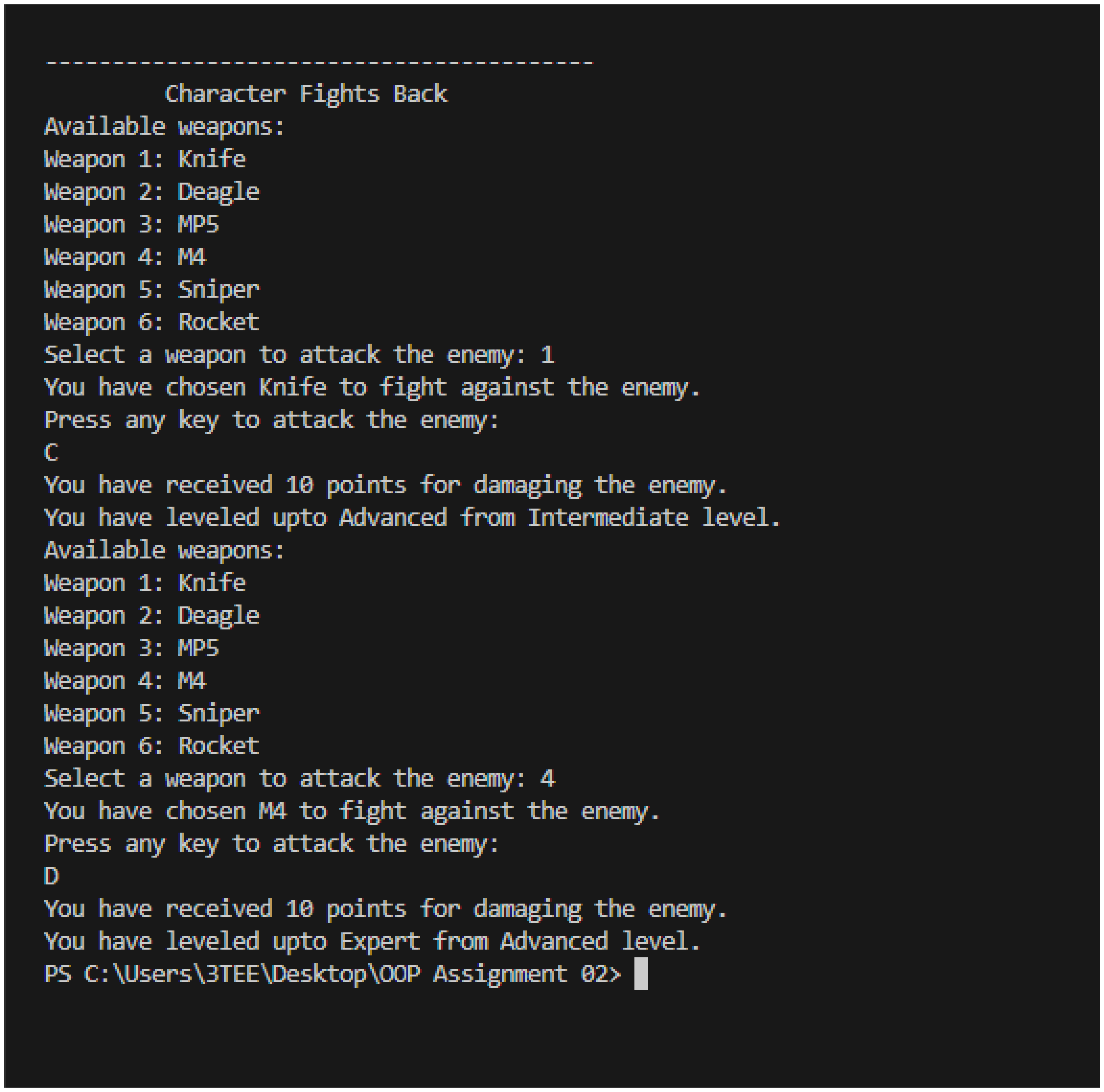
```cpp
void Character::levelUp () {
    if (points>=10) {
        level++;

        if (experience=="Beginner") {
            cout<<"You have leveled upto Intermediate from Beginner level." << endl;
            experience="Intermediate";
        }
        else if (experience=="Intermediate") {
            cout<<"You have leveled upto Advanced from Intermediate level." << endl;
            experience="Advanced";
        }
         else if (experience=="Advanced") {
            cout<<"You have leveled upto Expert from Advanced level." << endl;
            experience="Expert";
        }
         else {}
    }
}

void Character::playGame () {
    int choice;
    string startKey;
    vector <string> weapons = wep.getWeaponList();
    //vector <string> weapons;

    cout<<"Available weapons: " << endl;
    wep.displayWeapons();
    cout<< "Select a weapon to attack the enemy: ";
    cin>> choice;

    //weapons = getWeaponList();
    // weaponChar = weapons[choice-1]
    if (choice>(weapons.size()-1)) {
        cout<<"Invalid weapon choice. Try again." << endl;
        playGame();
    }
    else {
        weaponChar = weapons[choice-1];
        cout<<"You have chosen " << weaponChar << " to fight against the enemy." << endl;
        cout<<"Press any key to attack the enemy: " << endl;
        cin>> startKey;
        cout<<"You have received 10 points for damaging the enemy." << endl;
        points += 10;
        levelUp();
        enemy->deductHealth(5);
```



```cpp
void attack(Character &C, Enemy &E) {
    E.attack();
}

int main() {

    cout<<"*************************" << endl;
    cout<<"* Name: Muhammad Hammad *" << endl;
    cout<<"* Roll Number: 23K-2005 *" << endl;
    cout<<"*************************" << endl << endl;


    Character C(11, "Hammad");
    Enemy E(5); // sending 5 because mentioned in the question

    cout<<"--------------------------------------------------" << endl;
    cout<<"\t Adding Weapon" << endl;
    C.wep.addWeapon();

    cout<< endl <<"----------------------------------------" << endl;
    cout<<"\t Starting The Game" << endl;
    C.playGame();

    cout<< endl <<"----------------------------------------" << endl;
    cout<<"\t Enemy Attacks The Character" << endl;
    attack(C, E);

    cout<< endl <<"----------------------------------------" << endl;
    cout<<"\t Character Fights Back" << endl;
    C.playGame();
    C.playGame();

    return 0;
```



```
PS C:\Users\3TEE\Desktop\OOP Assignment 02> cd "c:\Users\3TEE\Desktop\OOP Assignment 02\"
*************************
* Name: Muhammad Hammad *
* Roll Number: 23K-2005 *
*************************


--------------------------------------------------
         Adding Weapon
Enter the number of weapons to add to the list: 1
Enter weapon 1 : Rocket

----------------------------------------
         Starting The Game
Available weapons:
Weapon 1: Knife
Weapon 2: Deagle
Weapon 3: MP5
Weapon 4: M4
Weapon 5: Sniper
Weapon 6: Rocket
Select a weapon to attack the enemy: 3
You have chosen MP5 to fight against the enemy.
Press any key to attack the enemy:
A
You have received 10 points for damaging the enemy.
You have leveled upto Intermediate from Beginner level.

----------------------------------------
         Enemy Attacks The Character
Available Weapons for Enemy:
Weapon 1: Knife
Weapon 2: Deagle
Weapon 3: MP5
Weapon 4: M4
Weapon 5: Sniper
Which weapon does the enemy want to choose? 2
You have chosenDeagle to attack the character.
Press any key to attack the character.
B
The character has lost 5 health.
```



```
-----------------------------------------
         Character Fights Back
Available weapons:
Weapon 1: Knife
Weapon 2: Deagle
Weapon 3: MP5
Weapon 4: M4
Weapon 5: Sniper
Weapon 6: Rocket
Select a weapon to attack the enemy: 1
You have chosen Knife to fight against the enemy.
Press any key to attack the enemy:
C
You have received 10 points for damaging the enemy.
You have leveled upto Advanced from Intermediate level.
Available weapons:
Weapon 1: Knife
Weapon 2: Deagle
Weapon 3: MP5
Weapon 4: M4
Weapon 5: Sniper
Weapon 6: Rocket
Select a weapon to attack the enemy: 4
You have chosen M4 to fight against the enemy.
Press any key to attack the enemy:
D
You have received 10 points for damaging the enemy.
You have leveled upto Expert from Advanced level.
PS C:\Users\3TEE\Desktop\OOP Assignment 02>
```

## Question 03:

### Task Three:

**Daraz Loyalty Program System**

In this scenario, Daraz is launching a loyalty program for its customers.

Design a class named DarazPersonData with the following member variables:

- lastName (string)
- firstName (string)
- address (string)
- city (string)
- state (string)
- zip (string)
- phone (string)
- Write the appropriate accessor and mutator functions for these member variables.

Next, design a class named DarazCustomerData. The DarazCustomerData class should have the following member variables:

- customerNumber (an int)
- loyaltyPoints (an int)

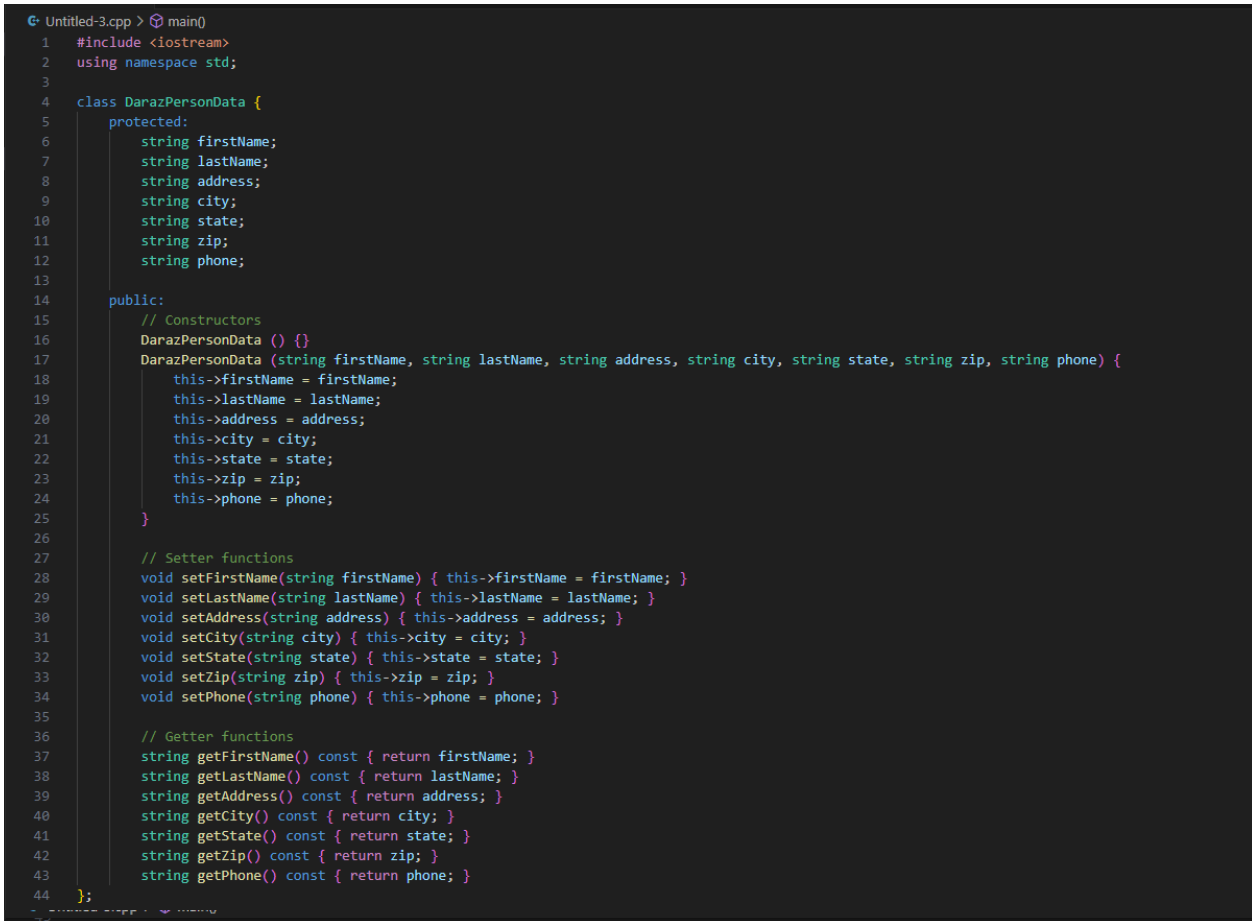
The customerNumber variable will hold a unique integer for each customer. The loyaltyPoints variable will track the loyalty points earned by the customer. Write appropriate accessor and mutator functions for these member variables.

Design a class named DarazLoyaltyProgram to manage the loyalty program:

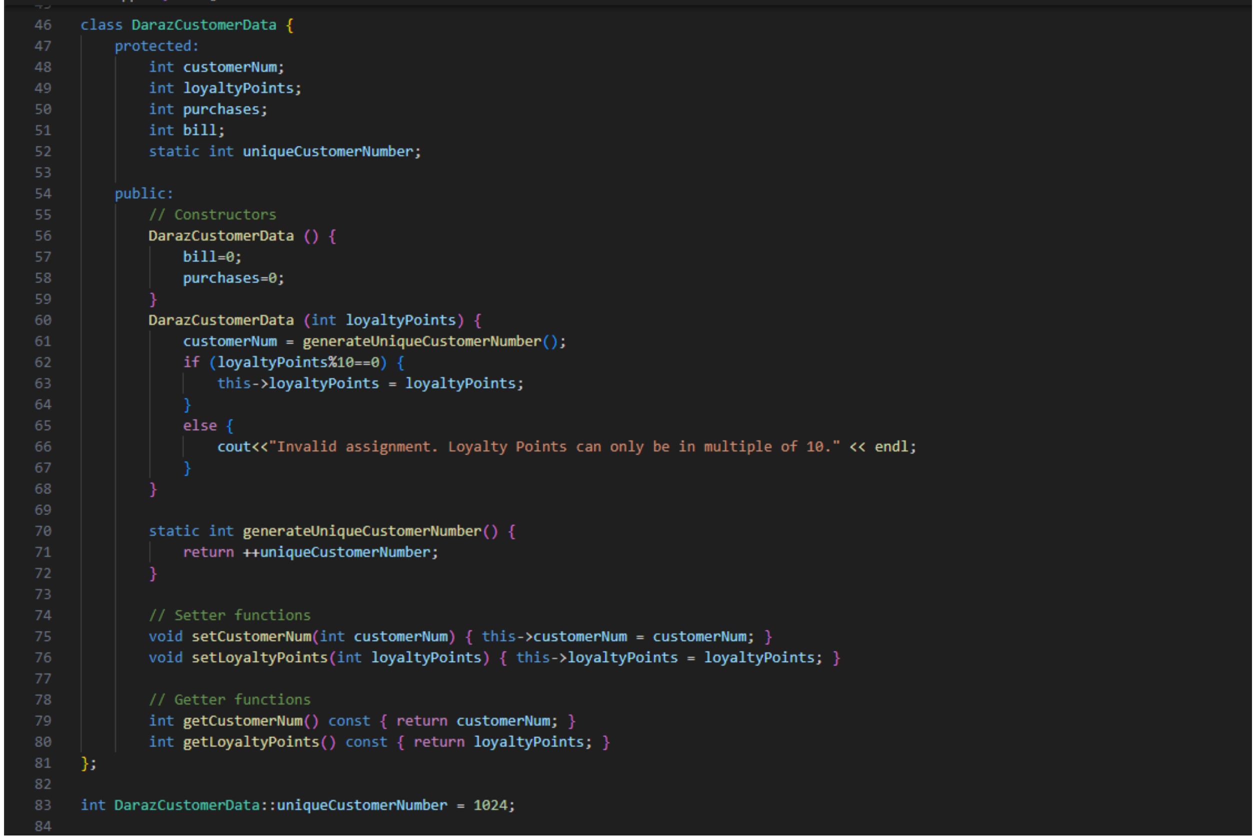
- Include functions to add loyalty points for purchases, redeem loyalty points for discounts, and display the total loyalty points for a customer.

Demonstrate the classes in a program by creating objects and performing operations such as adding loyalty points for purchases, redeeming loyalty points for discounts, and displaying total loyalty points for a customer.

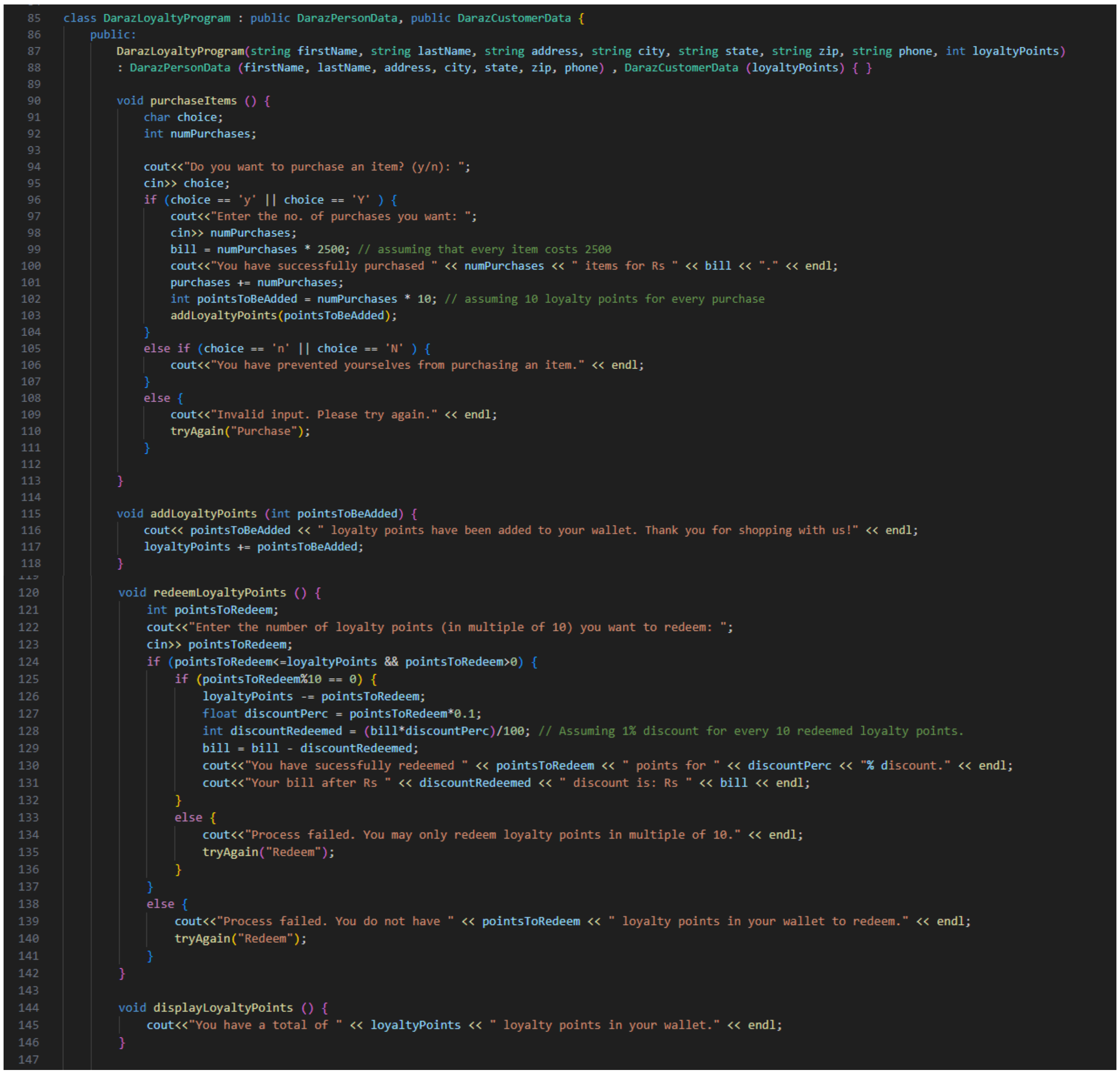
**Input Validation: Do not accept negative values for loyalty points or invalid customer numbers.**



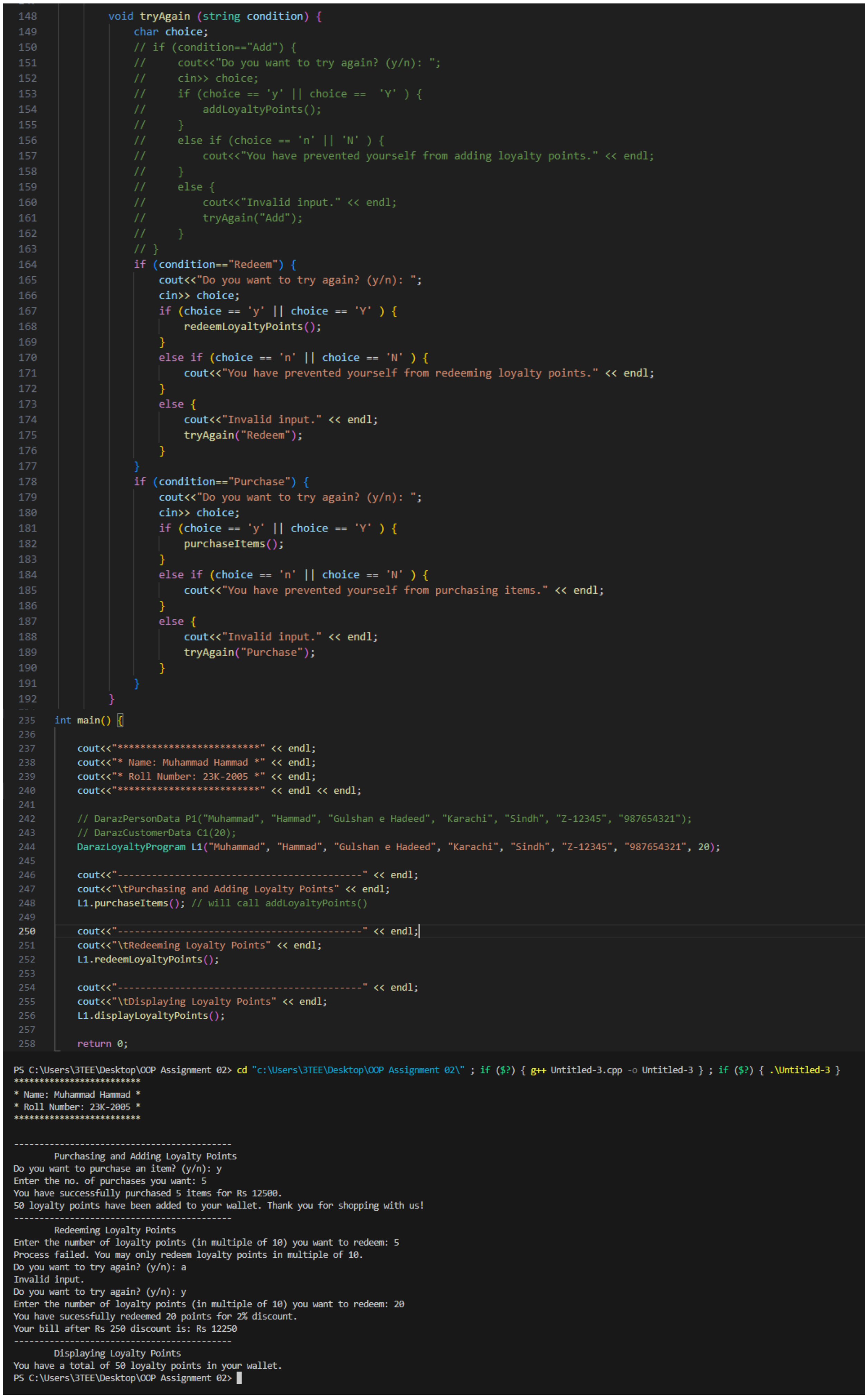
```cpp
#include <iostream>
using namespace std;

class DarazPersonData {
    protected:
        string firstName;
        string lastName;
        string address;
        string city;
        string state;
        string zip;
        string phone;

    public:
        // Constructors
        DarazPersonData () {}
        DarazPersonData (string firstName, string lastName, string address, string city, string state, string zip, string phone) {
            this->firstName = firstName;
            this->lastName = lastName;
            this->address = address;
            this->city = city;
            this->state = state;
            this->zip = zip;
            this->phone = phone;
        }

        // Setter functions
        void setFirstName(string firstName) { this->firstName = firstName; }
        void setLastName(string lastName) { this->lastName = lastName; }
        void setAddress(string address) { this->address = address; }
        void setCity(string city) { this->city = city; }
        void setState(string state) { this->state = state; }
        void setZip(string zip) { this->zip = zip; }
        void setPhone(string phone) { this->phone = phone; }

        // Getter functions
        string getFirstName() const { return firstName; }
        string getLastName() const { return lastName; }
        string getAddress() const { return address; }
        string getCity() const { return city; }
        string getState() const { return state; }
        string getZip() const { return zip; }
        string getPhone() const { return phone; }
};
```



```cpp
class DarazCustomerData {
    protected:
        int customerNum;
        int loyaltyPoints;
        int purchases;
        int bill;
        static int uniqueCustomerNumber;

    public:
        // Constructors
        DarazCustomerData () {
            bill=0;
            purchases=0;
        }
        DarazCustomerData (int loyaltyPoints) {
            customerNum = generateUniqueCustomerNumber();
            if (loyaltyPoints%10==0) {
                this->loyaltyPoints = loyaltyPoints;
            }
            else {
                cout<<"Invalid assignment. Loyalty Points can only be in multiple of 10." << endl;
            }
        }

        static int generateUniqueCustomerNumber() {
            return ++uniqueCustomerNumber;
        }

        // Setter functions
        void setCustomerNum(int customerNum) { this->customerNum = customerNum; }
        void setLoyaltyPoints(int loyaltyPoints) { this->loyaltyPoints = loyaltyPoints; }

        // Getter functions
        int getCustomerNum() const { return customerNum; }
        int getLoyaltyPoints() const { return loyaltyPoints; }
};

int DarazCustomerData::uniqueCustomerNumber = 1024;
```



```cpp
class DarazLoyaltyProgram : public DarazPersonData, public DarazCustomerData {
    public:
        DarazLoyaltyProgram(string firstName, string lastName, string address, string city, string state, string zip, string phone, int loyaltyPoints)
        : DarazPersonData (firstName, lastName, address, city, state, zip, phone) , DarazCustomerData (loyaltyPoints) { }

        void purchaseItems () {
            char choice;
            int numPurchases;

            cout<<"Do you want to purchase an item? (y/n): ";
            cin>> choice;
            if (choice == 'y' || choice == 'Y' ) {
                cout<<"Enter the no. of purchases you want: ";
                cin>> numPurchases;
                bill = numPurchases * 2500; // assuming that every item costs 2500
                cout<<"You have successfully purchased " << numPurchases << " items for Rs " << bill << "." << endl;
                purchases += numPurchases;
                int pointsToBeAdded = numPurchases * 10; // assuming 10 loyalty points for every purchase
                addLoyaltyPoints(pointsToBeAdded);
            }
            else if (choice == 'n' || choice == 'N' ) {
                cout<<"You have prevented yourselves from purchasing an item." << endl;
            }
            else {
                cout<<"Invalid input. Please try again." << endl;
                tryAgain("Purchase");
            }

        }

        void addLoyaltyPoints (int pointsToBeAdded) {
            cout<< pointsToBeAdded << " loyalty points have been added to your wallet. Thank you for shopping with us!" << endl;
            loyaltyPoints += pointsToBeAdded;
        }

        void redeemLoyaltyPoints () {
            int pointsToRedeem;
            cout<<"Enter the number of loyalty points (in multiple of 10) you want to redeem: ";
            cin>> pointsToRedeem;
            if (pointsToRedeem<=loyaltyPoints && pointsToRedeem>0) {
                if (pointsToRedeem%10 == 0) {
                    loyaltyPoints -= pointsToRedeem;
                    float discountPerc = pointsToRedeem*0.1;
                    int discountRedeemed = (bill*discountPerc)/100; // Assuming 1% discount for every 10 redeemed loyalty points.
                    bill = bill - discountRedeemed;
                    cout<<"You have sucessfully redeemed " << pointsToRedeem << " points for " << discountPerc << "% discount." << endl;
                    cout<<"Your bill after Rs " << discountRedeemed << " discount is: Rs " << bill << endl;
                }
                else {
                    cout<<"Process failed. You may only redeem loyalty points in multiple of 10." << endl;
                    tryAgain("Redeem");
                }
            }
            else {
                cout<<"Process failed. You do not have " << pointsToRedeem << " loyalty points in your wallet to redeem." << endl;
                tryAgain("Redeem");
            }
        }

        void displayLoyaltyPoints () {
            cout<<"You have a total of " << loyaltyPoints << " loyalty points in your wallet." << endl;
        }

```



```cpp
        void tryAgain (string condition) {
            char choice;
            // if (condition=="Add") {
            //     cout<<"Do you want to try again? (y/n): ";
            //     cin>> choice;
            //     if (choice == 'y' || choice ==  'Y' ) {
            //         addLoyaltyPoints();
            //     }
            //     else if (choice == 'n' || 'N' ) {
            //         cout<<"You have prevented yourself from adding loyalty points." << endl;
            //     }
            //     else {
            //         cout<<"Invalid input." << endl;
            //         tryAgain("Add");
            //     }
            // }
            if (condition=="Redeem") {
                cout<<"Do you want to try again? (y/n): ";
                cin>> choice;
                if (choice == 'y' || choice == 'Y' ) {
                    redeemLoyaltyPoints();
                }
                else if (choice == 'n' || choice == 'N' ) {
                    cout<<"You have prevented yourself from redeeming loyalty points." << endl;
                }
                else {
                    cout<<"Invalid input." << endl;
                    tryAgain("Redeem");
                }
            }
            if (condition=="Purchase") {
                cout<<"Do you want to try again? (y/n): ";
                cin>> choice;
                if (choice == 'y' || choice == 'Y' ) {
                    purchaseItems();
                }
                else if (choice == 'n' || choice == 'N' ) {
                    cout<<"You have prevented yourself from purchasing items." << endl;
                }
                else {
                    cout<<"Invalid input." << endl;
                    tryAgain("Purchase");
                }
            }
        }
int main() {

    cout<<"************************" << endl;
    cout<<"* Name: Muhammad Hammad *" << endl;
    cout<<"* Roll Number: 23K-2005 *" << endl;
    cout<<"************************" << endl << endl;

    // DarazPersonData P1("Muhammad", "Hammad", "Gulshan e Hadeed", "Karachi", "Sindh", "Z-12345", "987654321");
    // DarazCustomerData C1(20);
    DarazLoyaltyProgram L1("Muhammad", "Hammad", "Gulshan e Hadeed", "Karachi", "Sindh", "Z-12345", "987654321", 20);

    cout<<"------------------------------------------" << endl;
    cout<<"\tPurchasing and Adding Loyalty Points" << endl;
    L1.purchaseItems(); // will call addLoyaltyPoints()

    cout<<"------------------------------------------" << endl;
    cout<<"\tRedeeming Loyalty Points" << endl;
    L1.redeemLoyaltyPoints();

    cout<<"------------------------------------------" << endl;
    cout<<"\tDisplaying Loyalty Points" << endl;
    L1.displayLoyaltyPoints();

    return 0;
```

```
PS C:\Users\3TEE\Desktop\OOP Assignment 02> cd "c:\Users\3TEE\Desktop\OOP Assignment 02\" ; if ($?) { g++ Untitled-3.cpp -o Untitled-3 } ; if ($?) { .\Untitled-3 }
************************
* Name: Muhammad Hammad *
* Roll Number: 23K-2005 *
************************

------------------------------------------
        Purchasing and Adding Loyalty Points
Do you want to purchase an item? (y/n): y
Enter the no. of purchases you want: 5
You have successfully purchased 5 items for Rs 12500.
50 loyalty points have been added to your wallet. Thank you for shopping with us!
------------------------------------------
        Redeeming Loyalty Points
Enter the number of loyalty points (in multiple of 10) you want to redeem: 5
Process failed. You may only redeem loyalty points in multiple of 10.
Do you want to try again? (y/n): a
Invalid input.
Do you want to try again? (y/n): y
Enter the number of loyalty points (in multiple of 10) you want to redeem: 20
You have sucessfully redeemed 20 points for 2% discount.
Your bill after Rs 250 discount is: Rs 12250
------------------------------------------
        Displaying Loyalty Points
You have a total of 50 loyalty points in your wallet.
PS C:\Users\3TEE\Desktop\OOP Assignment 02>
```

## Question 04:

### Task Four:

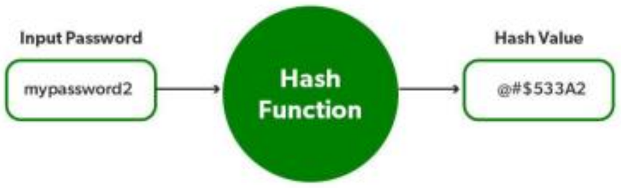
You've been tasked with designing the core components of a social media app similar to Instagram. The app will allow users to create profiles, post content, interact with posts (e.g., liking, commenting), and view their feed. There are different types of users, each with specific functionalities and access levels.

Tasks:

User Class Design:

- Design a base class User to represent common attributes and functionalities shared by all users, including username, email, and password.
- Implement user verification and password encryption to enhance security.

*[Choose a suitable encryption algorithm (e.g., bcrypt, Argon2) for securely hashing passwords.]*



Derived User Classes:

- Create derived classes for different types of users: RegularUser, and BusinessUser.
- Each derived class should inherit from the User class and provide specialized behavior based on the user's role and access level.
  - RegularUser Class:
    - Limited Posting: Regular users can post a maximum of 5 posts. Implement logic to enforce this limit.
    - Interactions: Regular users can like posts, comment on posts, and view their feed.
      - The RegularUser class maintains an array feed to store pointers to Post objects.
      - The addToFeed() method adds a post to the feed if there is space available.
      - The viewFeed() method displays the posts in the feed by iterating over the array and calling the display() method of each Post object.
      - Note: max feed size is 10; static const int MAX_FEED_SIZE = 10;
  - BusinessUser Class:
    - Post Promotion: Business users can promote their posts to reach a larger audience. Implement a method to promote posts.
      - User Validation: Ensure that only BusinessUser objects can invoke the promotePost() method.

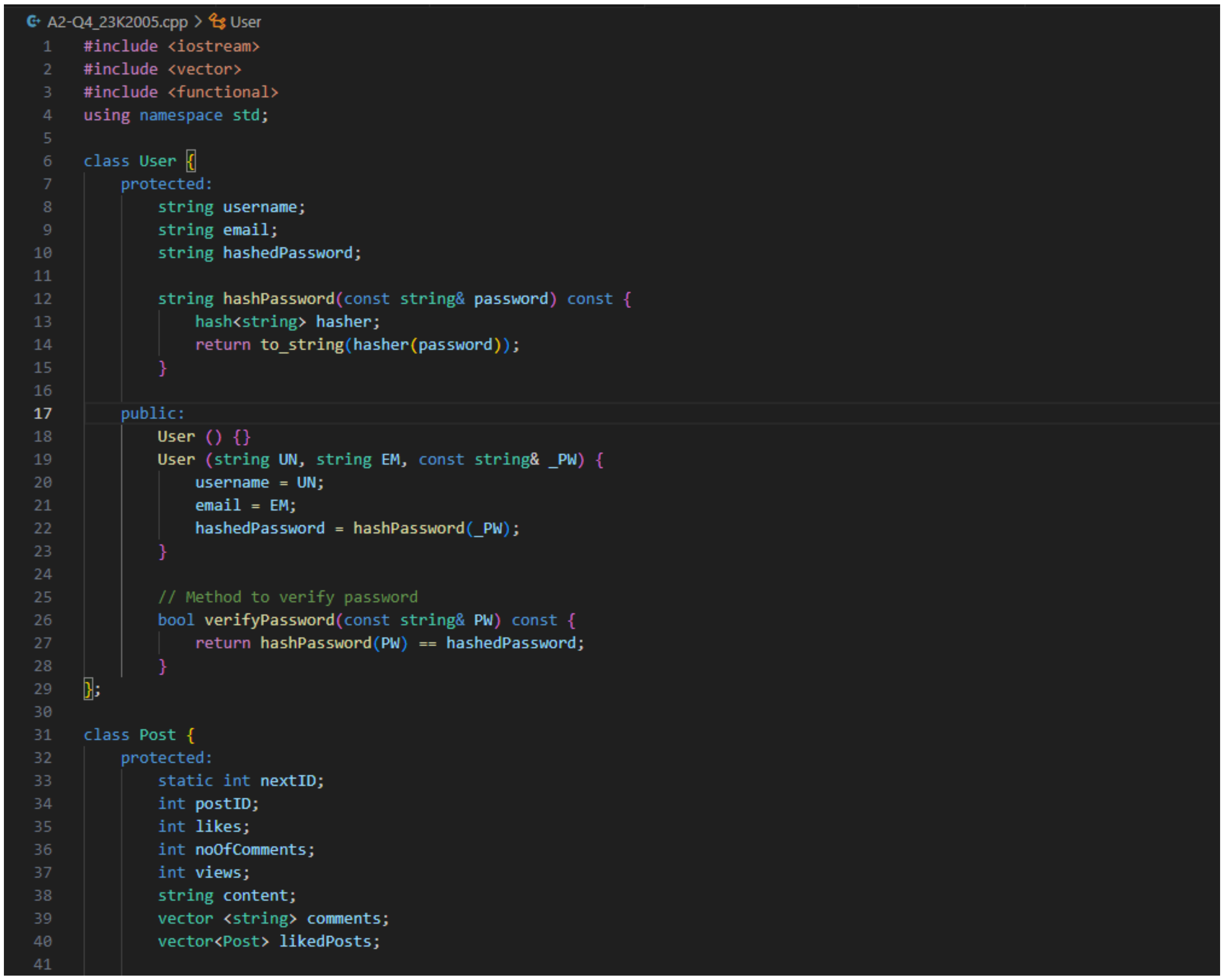
    - Promotion Limit: Apply a limit on the number of posts a business user can promote. [let's say 10 posts only]
    - Post Visibility: A custom logic within the promotePost() method to increase the post's likes by double and views by thrice.
  - Analytics Integration: Enhance the User and Post classes to include the following analytics functionalities:
    - Likes Tracking: Implement methods to track and retrieve the number of likes for each post.
    - Comments Tracking: Implement methods to track and retrieve the number of comments for each post.
    - Views Tracking: Implement methods to track and retrieve the number of views for each post.

<u>Post Class Design:</u>

- Define a class Post to represent individual posts in the app. Consider properties like postId, content, likes, comments, etc., and methods for adding comments, liking posts, and displaying post details.

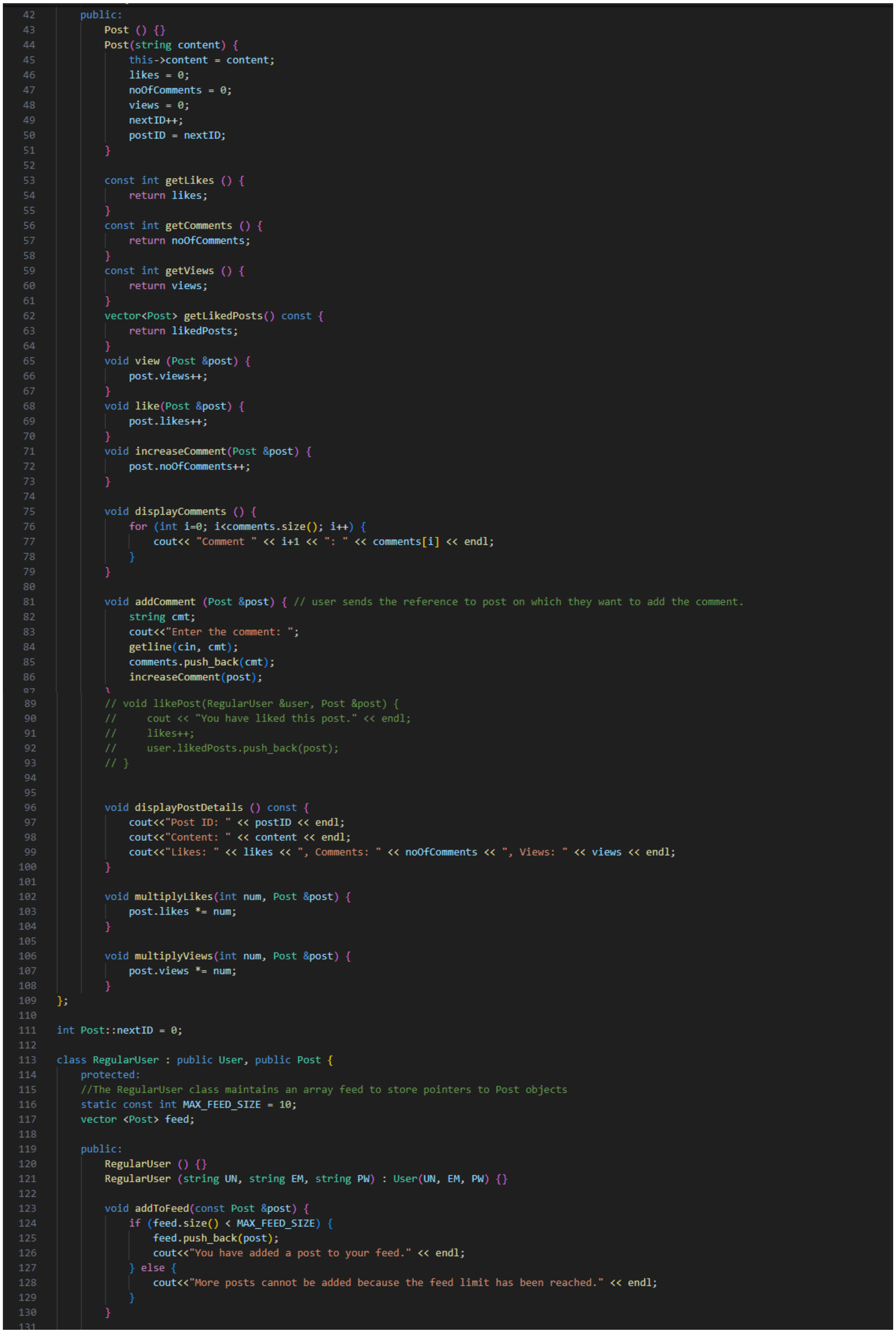
Interaction Simulation:

- Simulate interactions within the app by creating instances of different types of users and posts. Demonstrate how users can post content, interact with posts (e.g., liking, commenting), and view their feed. Use polymorphism to ensure that the same methods can be used uniformly across different user types.

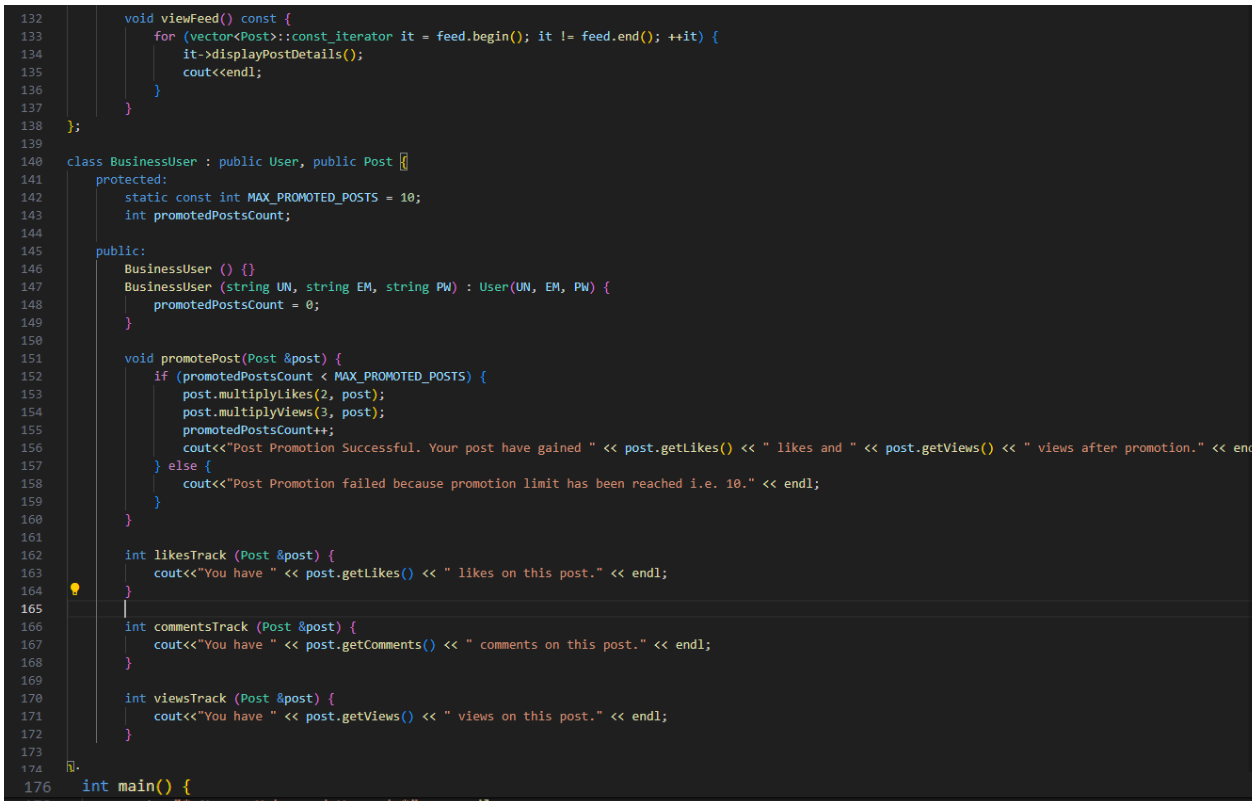


A2-Q4_23K2005.cpp > User

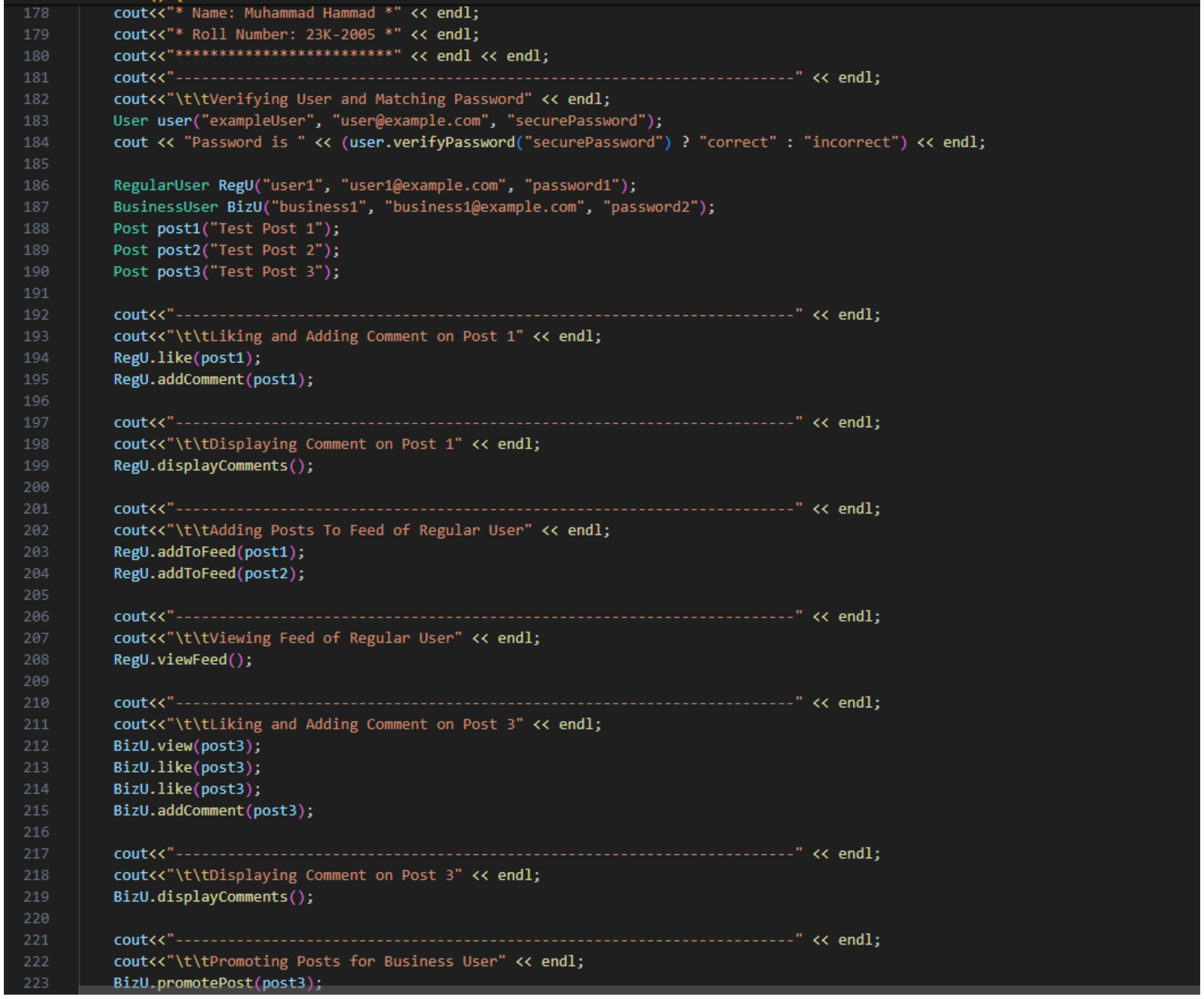
```cpp
#include <iostream>
#include <vector>
#include <functional>
using namespace std;

class User {
    protected:
        string username;
        string email;
        string hashedPassword;

        string hashPassword(const string& password) const {
            hash<string> hasher;
            return to_string(hasher(password));
        }

    public:
        User () {}
        User (string UN, string EM, const string& _PW) {
            username = UN;
            email = EM;
            hashedPassword = hashPassword(_PW);
        }

        // Method to verify password
        bool verifyPassword(const string& PW) const {
            return hashPassword(PW) == hashedPassword;
        }
};

class Post {
    protected:
        static int nextID;
        int postID;
        int likes;
        int noOfComments;
        int views;
        string content;
        vector <string> comments;
        vector<Post> likedPosts;

```



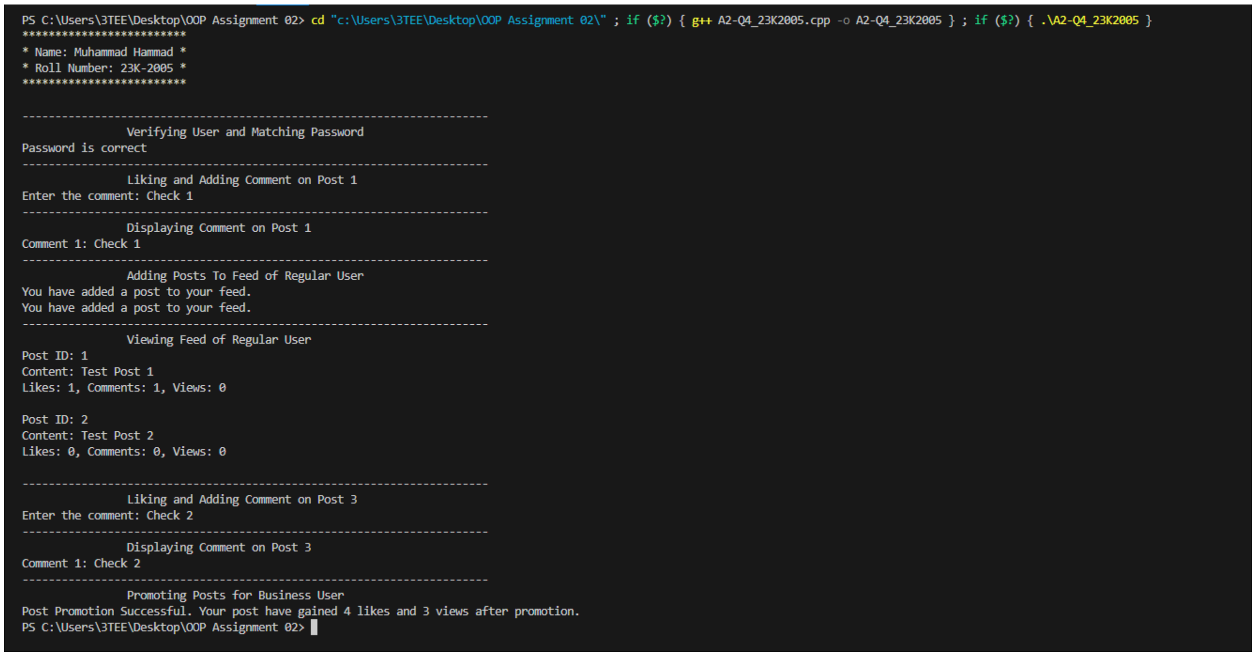
```cpp
    public:
        Post () {}
        Post(string content) {
            this->content = content;
            likes = 0;
            noOfComments = 0;
            views = 0;
            nextID++;
            postID = nextID;
        }

        const int getLikes () {
            return likes;
        }
        const int getComments () {
            return noOfComments;
        }
        const int getViews () {
            return views;
        }
        vector<Post> getLikedPosts() const {
            return likedPosts;
        }
        void view (Post &post) {
            post.views++;
        }
        void like(Post &post) {
            post.likes++;
        }
        void increaseComment(Post &post) {
            post.noOfComments++;
        }

        void displayComments () {
            for (int i=0; i<comments.size(); i++) {
                cout<< "Comment " << i+1 << ": " << comments[i] << endl;
            }
        }

        void addComment (Post &post) { // user sends the reference to post on which they want to add the comment.
            string cmt;
            cout<<"Enter the comment: ";
            getline(cin, cmt);
            comments.push_back(cmt);
            increaseComment(post);
        }
        // void likePost(RegularUser &user, Post &post) {
        //     cout << "You have liked this post." << endl;
        //     likes++;
        //     user.likedPosts.push_back(post);
        // }


        void displayPostDetails () const {
            cout<<"Post ID: " << postID << endl;
            cout<<"Content: " << content << endl;
            cout<<"Likes: " << likes << ", Comments: " << noOfComments << ", Views: " << views << endl;
        }

        void multiplyLikes(int num, Post &post) {
            post.likes *= num;
        }

        void multiplyViews(int num, Post &post) {
            post.views *= num;
        }
};

int Post::nextID = 0;

class RegularUser : public User, public Post {
    protected:
    //The RegularUser class maintains an array feed to store pointers to Post objects
    static const int MAX_FEED_SIZE = 10;
    vector <Post> feed;

    public:
        RegularUser () {}
        RegularUser (string UN, string EM, string PW) : User(UN, EM, PW) {}

        void addToFeed(const Post &post) {
            if (feed.size() < MAX_FEED_SIZE) {
                feed.push_back(post);
                cout<<"You have added a post to your feed." << endl;
            } else {
                cout<<"More posts cannot be added because the feed limit has been reached." << endl;
            }
        }
```



```cpp
        void viewFeed() const {
            for (vector<Post>::const_iterator it = feed.begin(); it != feed.end(); ++it) {
                it->displayPostDetails();
                cout<<endl;
            }
        }
};

class BusinessUser : public User, public Post {
    protected:
        static const int MAX_PROMOTED_POSTS = 10;
        int promotedPostsCount;

    public:
        BusinessUser () {}
        BusinessUser (string UN, string EM, string PW) : User(UN, EM, PW) {
            promotedPostsCount = 0;
        }

        void promotePost(Post &post) {
            if (promotedPostsCount < MAX_PROMOTED_POSTS) {
                post.multiplyLikes(2, post);
                post.multiplyViews(3, post);
                promotedPostsCount++;
                cout<<"Post Promotion Successful. Your post have gained " << post.getLikes() << " likes and " << post.getViews() << " views after promotion." << en
            } else {
                cout<<"Post Promotion failed because promotion limit has been reached i.e. 10." << endl;
            }
        }

        int likesTrack (Post &post) {
            cout<<"You have " << post.getLikes() << " likes on this post." << endl;
        }

        int commentsTrack (Post &post) {
            cout<<"You have " << post.getComments() << " comments on this post." << endl;
        }

        int viewsTrack (Post &post) {
            cout<<"You have " << post.getViews() << " views on this post." << endl;
        }

}.
```



```cpp
int main() {
    cout<<"* Name: Muhammad Hammad *" << endl;
    cout<<"* Roll Number: 23K-2005 *" << endl;
    cout<<"************************" << endl << endl;
    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tVerifying User and Matching Password" << endl;
    User user("exampleUser", "user@example.com", "securePassword");
    cout << "Password is " << (user.verifyPassword("securePassword") ? "correct" : "incorrect") << endl;

    RegularUser RegU("user1", "user1@example.com", "password1");
    BusinessUser BizU("business1", "business1@example.com", "password2");
    Post post1("Test Post 1");
    Post post2("Test Post 2");
    Post post3("Test Post 3");

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tLiking and Adding Comment on Post 1" << endl;
    RegU.like(post1);
    RegU.addComment(post1);

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tDisplaying Comment on Post 1" << endl;
    RegU.displayComments();

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tAdding Posts To Feed of Regular User" << endl;
    RegU.addToFeed(post1);
    RegU.addToFeed(post2);

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tViewing Feed of Regular User" << endl;
    RegU.viewFeed();

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tLiking and Adding Comment on Post 3" << endl;
    BizU.view(post3);
    BizU.like(post3);
    BizU.like(post3);
    BizU.addComment(post3);

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tDisplaying Comment on Post 3" << endl;
    BizU.displayComments();

    cout<<"------------------------------------------------------------------------" << endl;
    cout<<"\t\tPromoting Posts for Business User" << endl;
    BizU.promotePost(post3);
```



```
PS C:\Users\3TEE\Desktop\OOP Assignment 02> cd "c:\Users\3TEE\Desktop\OOP Assignment 02\" ; if ($?) { g++ A2-Q4_23K2005.cpp -o A2-Q4_23K2005 } ; if ($?) { .\A2-Q4_23K2005 }
*************************
* Name: Muhammad Hammad *
* Roll Number: 23K-2005 *
*************************

-----------------------------------------------------------------------
                Verifying User and Matching Password
Password is correct
-----------------------------------------------------------------------
                Liking and Adding Comment on Post 1
Enter the comment: Check 1
-----------------------------------------------------------------------
                Displaying Comment on Post 1
Comment 1: Check 1
-----------------------------------------------------------------------
                Adding Posts To Feed of Regular User
You have added a post to your feed.
You have added a post to your feed.
-----------------------------------------------------------------------
                Viewing Feed of Regular User
Post ID: 1
Content: Test Post 1
Likes: 1, Comments: 1, Views: 0

Post ID: 2
Content: Test Post 2
Likes: 0, Comments: 0, Views: 0

-----------------------------------------------------------------------
                Liking and Adding Comment on Post 3
Enter the comment: Check 2
-----------------------------------------------------------------------
                Displaying Comment on Post 3
Comment 1: Check 2
-----------------------------------------------------------------------
                Promoting Posts for Business User
Post Promotion Successful. Your post have gained 4 likes and 3 views after promotion.
PS C:\Users\3TEE\Desktop\OOP Assignment 02> 
```

**Note:** I have used multiple inheritance in Question 4 because there was no restriction about it.