ASSIGNMENT 2

***QUESTION 1***

**CODE**

#include <iostream>

using namespace std;

class SecurityTool

{

protected:

string securityLevel;

int cost;

int noOfDevices;

public:

string GetSecurityLevel() const

{

return securityLevel;

}

void SetSecurityLevel(string \_securityLevel)

{

checkSecurityLevel(\_securityLevel);

}

int GetCost() const

{

return cost;

}

void SetCost(int \_cost)

{

checkCost(\_cost);

}

int GetNoOfDevices() const

{

return noOfDevices;

}

public:

SecurityTool(string \_SecurityLevel, int \_cost, int \_noOfDevices)

: noOfDevices(\_noOfDevices)

{

checkCost(\_cost);

checkSecurityLevel(\_SecurityLevel);

}

void performScan()

{

cout << "Performing a generic security scan" << endl;

}

// the following functions are used to check the security level and cost, they are not required but were reused in constructor as wel as in setters

void checkSecurityLevel(string &\_SecurityLevel)

{

while (\_SecurityLevel != "High" && \_SecurityLevel != "Low" && \_SecurityLevel != "Medium")

{

cout << "please enter correct security level i.e 'High', 'Medium' or 'Low' : " << endl;

cin >> \_SecurityLevel;

}

securityLevel = \_SecurityLevel;

}

void checkCost(int &\_cost)

{

while (\_cost <= 0)

{

cout << "cost can't be less than or equal to 0, please enter correct cost: " << endl;

cin >> \_cost;

}

cost = \_cost;

}

};

class FirewallTool : public SecurityTool

{

protected:

int ports[23];

string Protocols[6];

public:

FirewallTool(string \_SecurityLevel, int \_cost, int id)

: SecurityTool(\_SecurityLevel, \_cost, 10)

{

generateList(id);

Protocols[0] = "HTTPS";

Protocols[1] = "FTP";

Protocols[2] = "UDP";

Protocols[3] = "ICMP";

Protocols[4] = "SSH";

Protocols[5] = "SNMP";

}

int returnNonZero(int id)

{

if (id % 10 != 0)

{

return (id % 10);

}

else if (id == 0)

{

cout << "\n Entered id contains only zeros";

return 0;

}

else

{

returnNonZero(id / 10);

}

}

void generateList(int id)

{

int n = returnNonZero(id); //mine id is 0800

for (int i = 0; i < 23; i++)

{

ports[i] = ++n;

}

}

void performScan()

{

SecurityTool::performScan();

if (securityLevel == "High")

{

cout << "Traffic from the port and protocol list are allowed, following are the ports, and the port list : " << endl;

for (int i = 0; i < 23; i++)

{

cout << ports[i] << " ";

}

cout << endl;

for (int i = 0; i < 6; i++)

{

cout << Protocols[i] << endl;

}

}

else if (securityLevel == "Medium")

{

cout << "Traffic from the port along with the next two ports and protocol list are allowed, following are the ports, and the port list : " << endl;

for (int i = 0; i < 23; i++)

{

cout << ports[i] << " ";

}

int n = ports[22];

cout << ++n << " ";

cout << ++n << endl;

for (int i = 0; i < 6; i++)

{

cout << Protocols[i] << endl;

}

}

else

{

cout << "Traffic from the port along with the next 5 ports and from protocol list along with TCP and DNS are allowed, following are the ports, and the port list : " << endl;

for (int i = 0; i < 23; i++)

{

cout << ports[i] << " ";

}

int n = ports[22];

for (int i = 0; i < 5; ++i)

{

cout << ++n << " ";

}

cout << endl;

for (int i = 0; i < 6; i++)

{

cout << Protocols[i] << endl;

}

cout << "TCP" << endl;

cout << "DNS" << endl;

}

}

};

int main()

{

cout << "Muhammad mufeez 23k0800";

FirewallTool f1("High", 100,800);

f1.performScan();

FirewallTool f2("Low", 0,9087);

f2.performScan();

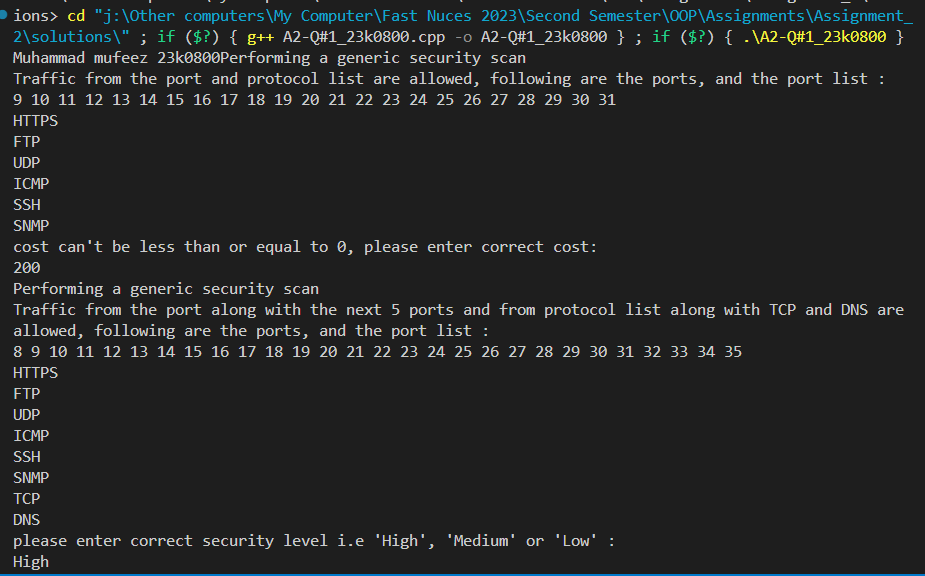
FirewallTool f3("Med", 23333,2000);

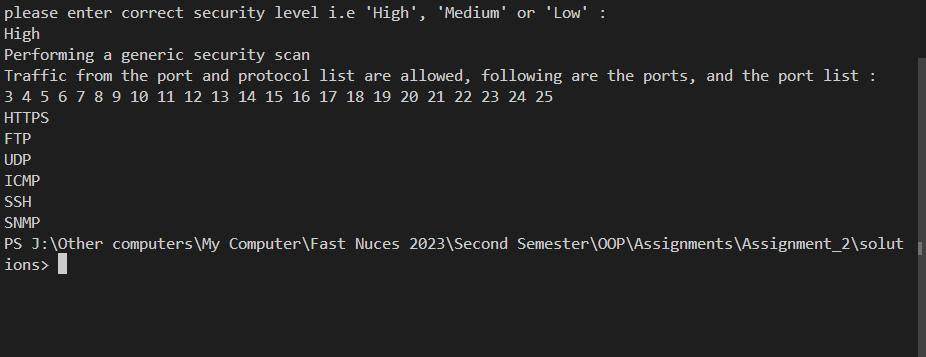
f3.performScan();

return 0;

}

**OUTPUT**





***QUESTION 2***

**CODE**

#include <iostream>

using namespace std;

class Enemy;

class Character;

class Player;

class Weapon;

class Player

{

protected:

int playerId;

string playerName;

int health;

public:

Player(int \_playerId, string \_playerName) : playerId(\_playerId), playerName(\_playerName), health(100) {}

int GetPlayerId() const

{

return playerId;

}

void setHealth(int health)

{

this->health = health;

cout << "Remaining Health: " << this->health << endl;

}

string GetplayerName() const

{

return playerName;

}

int GetHealth() const

{

return this->health;

}

};

class Weapon

{

public:

string weaponList[20]; // we have max of 20 weapons

Weapon()

{

cout << "Start Entering weapons " << endl;

for (int i = 0; i < 20; i++)

{

getline(cin, weaponList[i]);

// cin.ignore();

if (weaponList[i] == "" && i < 5)

{

cout << "Enter atleast 5 weapons" << endl;

i--;

}

else if (weaponList[i] == "")

{

break;

}

}

}

void useWeapon()

{

string temp;

cout << "Enter name of weapon you wanna use: " << endl;

cin >> temp;

int i = 0;

while (weaponList[i] != "")

{

if (weaponList[i].compare(temp) == 0)

{

cout << weaponList[i] << " selected" << endl;

for (int j = i; j < 20 && weaponList[j] != ""; j++)

{

weaponList[j] = weaponList[j + 1];

}

break;

}

i++;

}

if (i == 0 && weaponList[i] == "")

{

cout << "No weapons left, you can't attack the enemy" << endl;

}

else if (i != 0 && weaponList[i - 1] == "")

{

cout << "Weapon not found" << endl;

useWeapon();

}

}

};

class Character : public Player, public Weapon

{

private:

int level;

string experience;

int points;

public:

Character(int \_playerId, string \_playerName) : Player(\_playerId, \_playerName), level(0), experience("beginner"), points(0) { cout << "New Player Added" << endl; }

void levelUp()

{

level++;

if (experience == "beginner")

{

experience = "intermediate";

}

else if (experience == "intermediate")

{

experience = "Advanced";

}

else if (experience == "Advanced")

{

experience = "Expert";

}

}

void PlayGame()

{

cout << "Player turn" << endl;

useWeapon();

points += 10;

levelUp();

cout << "Points: " << points << " Experience: " << experience << " Level: " << level << endl;

}

};

class Enemy : public Weapon, public Player

{

private:

int damage;

public:

Enemy(int damage, int \_playerId, string \_playerName) : Player(\_playerId, \_playerName), damage(damage)

{

while (damage < 1 || damage > 10)

{

cout << "Enter damage between 1 and 10" << endl;

cin >> damage;

}

cout << "New Enemy Added" << endl;

}

void attack(Character &player)

{

cout << "Enemy Turn" << endl;

useWeapon();

cout << "Attacked the player " << endl;

player.setHealth(player.GetHealth() - damage);

}

};

int main()

{

cout << "Muhammad mufeez 23k0800" << endl;

Character c1(23434, "Mufeez");

Enemy e1(10, 343, "Someone");

int choice;

do

{

cout << "Enter 1 to attack enemy, 2 to attack player, 0 to exit" << endl;

cin >> choice;

switch (choice)

{

case 1:

{

c1.PlayGame();

break;

}

case 2:

{

e1.attack(c1);

break;

}

case 0:

break;

default:

break;

}

} while (choice != 0 && c1.GetHealth() > 0);

if (c1.GetHealth() <= 0)

{

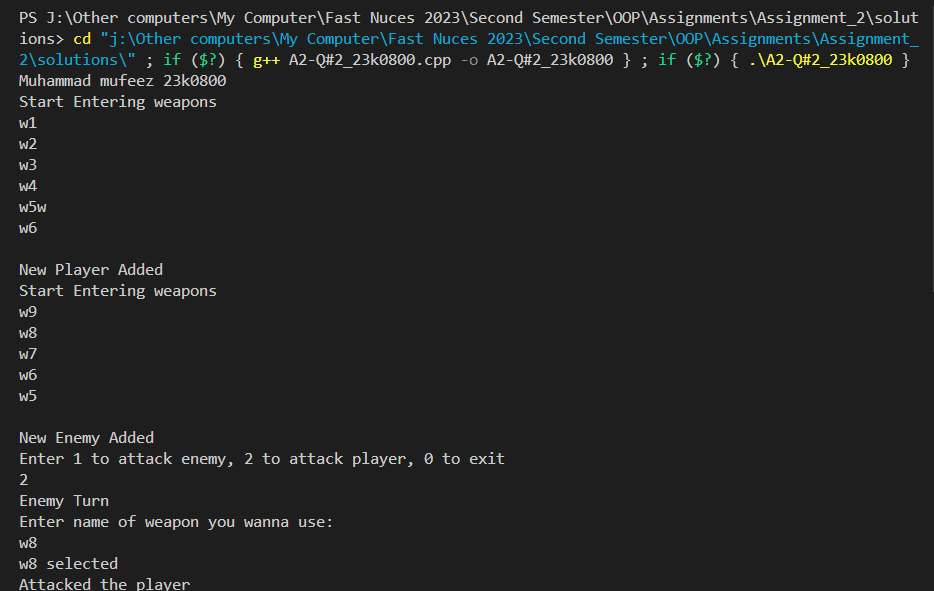
cout << "Player died, game over, restart the program to start a new game" << endl;

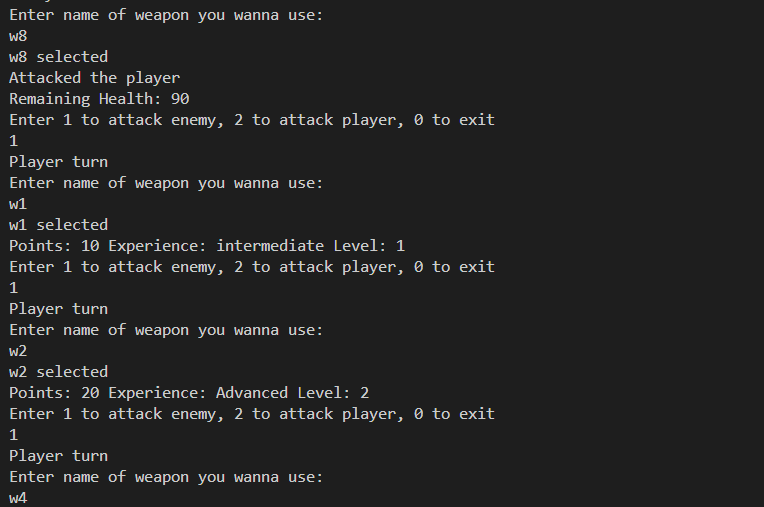
}

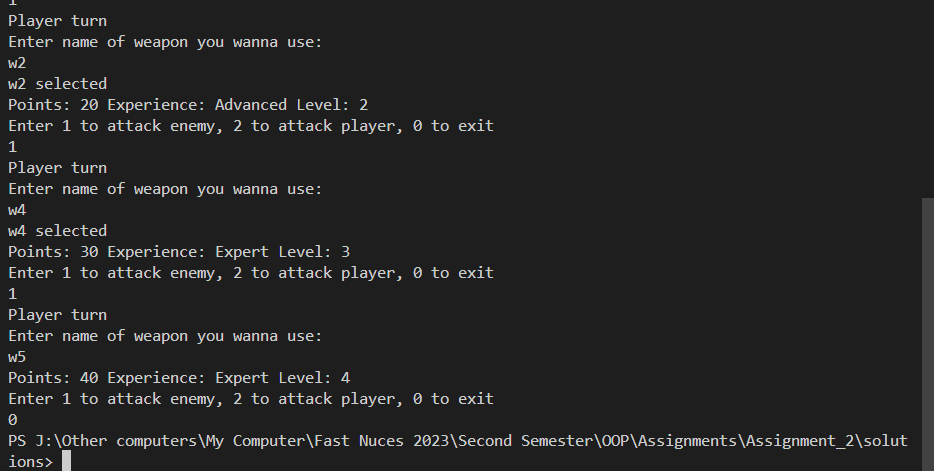
return 0;

}

**OUTPUT**







***QUESTION 3***

**CODE**

#include <iostream>

#include <vector>

using namespace std;

class DarazPersonData

{

private:

string firstName;

string lastName;

string address;

string city;

string state;

string zip;

string phone;

public:

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;

}

string GetFirstName() const

{

return firstName;

}

void SetFirstName(string firstName)

{

this->firstName = firstName;

}

string GetLastName() const

{

return lastName;

}

void SetLastName(string lastName)

{

this->lastName = lastName;

}

string GetAddress() const

{

return address;

}

void SetAddress(string address)

{

this->address = address;

}

string GetCity() const

{

return city;

}

void SetCity(string city)

{

this->city = city;

}

string GetState() const

{

return state;

}

void SetState(string state)

{

this->state = state;

}

string GetZip() const

{

return zip;

}

void SetZip(string zip)

{

this->zip = zip;

}

string GetPhone() const

{

return phone;

}

void SetPhone(string phone)

{

this->phone = phone;

}

};

class DarazCustomerData : public DarazPersonData

{

protected:

const int customerNumber;

int loyaltyPoints;

static int n;

public:

DarazCustomerData(string firstName, string lastName, string address, string city, string state, string zip, string phone) : DarazPersonData(firstName, lastName, address, city, state, zip, phone), customerNumber(n++), loyaltyPoints(0) {}

void setLoyaltyPoints(int loyaltyPoints) { this->loyaltyPoints = loyaltyPoints; }

int const GetcustomerNumber() const

{

return customerNumber;

}

int GetLoyaltyPoints() const

{

return loyaltyPoints;

}

};

int DarazCustomerData::n = 1;

class DarazLoyaltyProgram

{

public:

vector<DarazCustomerData> Customers;

int temp;

public:

void addNewCustomer(DarazCustomerData &c)

{

Customers.push\_back(c);

cout<<c.GetLastName()<<" has been added to the program successfully" << endl;

}

void addLoyaltyPoints(int cN, int points)

{

while (points < 0)

{

cout << "Enter valid points, i.e positive values" << endl;

cin >> points;

}

for (int i = 0; i < Customers.size(); i++)

{

if (Customers[i].GetcustomerNumber() == cN)

{

temp = Customers[i].GetLoyaltyPoints();

temp += points;

Customers[i].setLoyaltyPoints(temp);

cout << "Loyalty points added successfully" << endl;

return;

}

}

cout << "Customer not found" << endl;

}

void redeemLoyaltyPoints(int cN, int points)

{

while (points < 0)

{

cout << "Enter valid points, i.e positive values" << endl;

cin >> points;

}

for (int i = 0; i < Customers.size(); i++)

{

if (Customers[i].GetcustomerNumber() == cN)

{

temp = Customers[i].GetLoyaltyPoints();

temp -= points;

Customers[i].setLoyaltyPoints(temp);

cout << "Loyalty points redeemed successfully" << endl;

return;

}

}

cout << "Customer not found" << endl;

}

void displayLoyalPoints(int cN)

{

for (int i = 0; i < Customers.size(); i++)

{

if (Customers[i].GetcustomerNumber() == cN)

{

cout << "Total Points " << Customers[i].GetLoyaltyPoints() << endl;

return;

}

}

cout << "Customer not found" << endl;

}

};

int main()

{

cout << "Muhammad mufeez 23k0800" << endl;

DarazCustomerData c1("Muhammad", "Mufeez", "123 Main Street", "Anytown", "CA", "12345", "123-456-7890");

DarazCustomerData c2("John", "Doe", "456 Oak Street", "Anytown", "CA", "67890", "987-654-3210");

DarazCustomerData c3("Jane", "Doe", "789 Elm Street", "Anytown", "CA", "23456", "555-555-5555");

DarazLoyaltyProgram program;

program.addNewCustomer(c1);

program.addNewCustomer(c2);

program.addNewCustomer(c3);

program.addLoyaltyPoints(1, 100);

program.addLoyaltyPoints(2, 200);

program.addLoyaltyPoints(3, 300);

program.displayLoyalPoints(1);

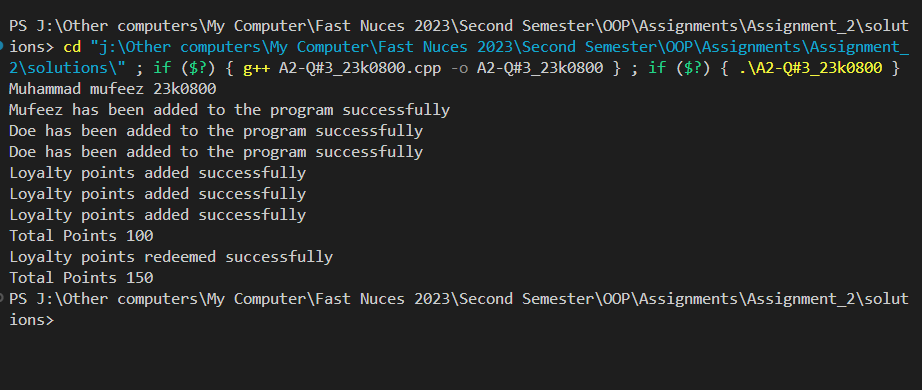
program.redeemLoyaltyPoints(2, 50);

program.displayLoyalPoints(2);

return 0;

}

**OUTPUT**



***QUESTION 4***

**CODE**

#include <iostream>

#include <vector>

using namespace std;

class User;

class Post

{

public:

static int postIdGlobal;

private:

string content;

string comments[100];

int commented;

int likes;

bool isPromoted;

int views;

int postId;

public:

Post(string \_content) : content(\_content), likes(0), views(0), isPromoted(false), commented(0) { postId = ++postIdGlobal; }

void setPromoted(bool \_promted) { isPromoted = \_promted; }

bool getPromoted() { return isPromoted; }

void editContent()

{ // extra method

cout << "Previous Content: " << endl;

cout << content << endl;

cout << "Enter new content: ";

getline(cin, content);

cin.ignore();

}

void addComment()

{

cout << "Enter comment: ";

string comment;

getline(cin, comment);

cin.ignore();

comments[commented++] = comment;

}

void addlike()

{

if (isPromoted)

{

likes += 2;

}

else

{

likes++;

}

}

void addView()

{

if (isPromoted)

{

views += 2;

}

else

{

views++;

}

}

void displayPost() const

{

cout << "Post ID: " << postId << endl;

cout << "Content: " << content << endl;

cout << "Likes: " << likes << endl;

cout << "Views: " << views << endl;

cout << "Comments: " << endl;

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

{

cout << comments[i] << endl;

}

}

};

int Post::postIdGlobal = 3434;

class User

{

private:

string email;

string username;

string password;

int userId;

public:

static int userIdGlobal;

public:

User(string \_email, string \_username, string \_password) : email(\_email), username(\_username)

{

userId = ++userIdGlobal;

passwordEncryption(\_password);

}

void passwordEncryption(string \_password)

{

string key = "secret";

for (int i = 0; i < \_password.length(); i++)

{ // XOREncryptDecrypt

\_password[i] = \_password[i] ^ key[i % key.length()];

}

password = \_password;

}

void likePost(Post &post)

{

post.addlike();

}

void commentPost(Post &post)

{

post.addComment();

}

};

int User::userIdGlobal = 234;

class RegularUser : public User

{

private:

int postLimit;

static const int MAX\_FEED\_SIZE;

Post \*feed[10];

int posted;

public:

RegularUser(string \_email, string \_username, string \_password) : User(\_email, \_username, \_password), postLimit(5), posted(0) {}

void addToFeed(Post &post)

{

if (posted < MAX\_FEED\_SIZE)

{

feed[posted++] = &post;

}

else

{

cout << "Feed is full. Cannot add post." << endl;

}

}

void viewFeed()

{

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

{

feed[i]->displayPost();

feed[i]->addView();

}

}

};

const int RegularUser::MAX\_FEED\_SIZE = 10;

class BusinessUserClass : public User

{

private:

const int limitPromoted;

int promotedPosts;

public:

BusinessUserClass(string \_email, string \_username, string \_password) : User(\_email, \_username, \_password), limitPromoted(10), promotedPosts(0) {}

void promotePost(Post &p)

{

if (promotedPosts < limitPromoted && !(p.getPromoted()))

{

p.setPromoted(true);

promotedPosts++;

cout << "Congratulations, post has been promoted successfully" << endl;

}

else if (p.getPromoted())

{

cout << "Post is already promoted" << endl;

}

else

{

cout << "You can't promote more posts, you have reached the max ammount to promote post" << endl;

}

}

void demotePost(Post &p)

{

if (p.getPromoted())

{

p.setPromoted(false);

promotedPosts--;

cout << "Post has been demoted successfully" << endl;

}

else

{

cout << "Post is already demoted" << endl;

}

}

};

int main()

{

cout << "Muhammad mufeez 23k0800" << endl;

RegularUser u1("u1@u1.com", "u1", "hdgiysdgsd");

BusinessUserClass b1("sdsddf@ds", "b1", "b1");

Post p1("This is my first post");

Post p2("This is my second post");

u1.addToFeed(p1);

u1.likePost(p1);

u1.commentPost(p1);

u1.viewFeed();

u1.addToFeed(p2);

u1.viewFeed();

b1.promotePost(p1);

b1.demotePost(p2);

b1.likePost(p1);

return 0;

}

**OUTPUT**

