**OOP Theory Assignmnent 2**

**23K0703**

**BCS-2D**

Question 1

Code:

**#**include <iostream>

using namespace std;

class SecurityTool

{

protected:

string securityLevel;

int numOfDevices;

float cost;

public:

SecurityTool(

string securityLevel,

int numOfDevices,

float cost)

{

if (securityLevel == "Low" || securityLevel == "Medium" || securityLevel == "High")

{

this->securityLevel = securityLevel;

}

if (cost <= 0)

cost = 1;

this->cost = cost;

}

bool performScan()

{

return false;

}

};

class FirewallTool : public SecurityTool

{

int ports[23];

string protocols[6];

public:

FirewallTool(

string securityLevel,

float cost,

int numOfDevices)

: SecurityTool(

securityLevel,

numOfDevices,

cost)

{

generateList();

}

void generateList()

{

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

{

ports[i] = i + 7 + 1;

}

protocols[0] = "HTTPS";

protocols[1] = "FTP";

protocols[2] = "UDP";

protocols[3] = "ICMP";

protocols[4] = "SSH";

protocols[5] = "SNMP";

}

bool performScan(int trafficPort, string trafficProtocol)

{

bool validPort = false, validProtocol = false;

if (securityLevel == "High")

{

// Checking if port is valid

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

{

if (ports[i] == trafficPort)

{

validPort = true;

break;

}

}

// Checking if protocol is valid

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

{

if (protocols[i] == trafficProtocol)

{

validProtocol = true;

break;

}

}

}

else if (securityLevel == "Medium")

{

// Checking for listed ports

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

{

if (ports[i] == trafficPort)

{

validPort = true;

break;

}

}

if (!validPort)

{

// Checking for 2 extra ports

for (int i = 1; i <= 2; i++)

{

if ((ports[19] + i) == trafficPort)

{

validPort = true;

break;

}

}

}

// Checking if protocol is valid

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

{

if (protocols[i] == trafficProtocol)

{

validProtocol = true;

break;

}

}

}

else if (securityLevel == "Low")

{

// Checking for listed ports

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

{

if (ports[i] == trafficPort)

{

validPort = true;

break;

}

}

if (!validPort)

{

// Checking for 5 extra ports

for (int i = 1; i <= 5; i++)

{

if ((ports[19] + i) == trafficPort)

{

validPort = true;

break;

}

}

}

// Checking if protocol is valid

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

{

if (protocols[i] == trafficProtocol)

{

validProtocol = true;

break;

}

}

if (!validProtocol)

{

validProtocol = trafficProtocol == "TCP" || trafficProtocol == "DNS";

}

}

return validPort && validProtocol;

}

};

int main()

{

// Solution Header

cout << "Name: Sarim Ahmed" << endl;

cout << "ID: 23K0703" << endl

<< endl;

string securityLevels[3] = {"High", "Medium", "Low"};

int securityLevelChoice;

cout << "Choose your security level" << endl;

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

{

cout << "[" << i << "] " << securityLevels[i] << endl;

}

cout << ": ";

cin >> securityLevelChoice;

FirewallTool firewallTool(securityLevels[securityLevelChoice], 13.0, 5);

string protocols[8] = {

"HTTPS",

"FTP",

"UDP",

"ICMP",

"SSH",

"SNMP",

"TCP",

"DNS",

};

string trafficProtocol;

int trafficPort, protocolChoice;

// Show all protocol options

cout << "Protocols;" << endl;

for (int i = 0; i < 8; i++)

{

cout << "[" << i << "] " << protocols[i] << endl;

}

cout << "Enter your protocol selection [0-7]: ";

cin >> protocolChoice;

cout << endl

<< "Enter your port number: ";

cin >> trafficPort;

bool allowed = firewallTool.performScan(trafficPort, protocols[protocolChoice]);

if (allowed)

{

cout << "Connection Allowed" << endl;

}

else

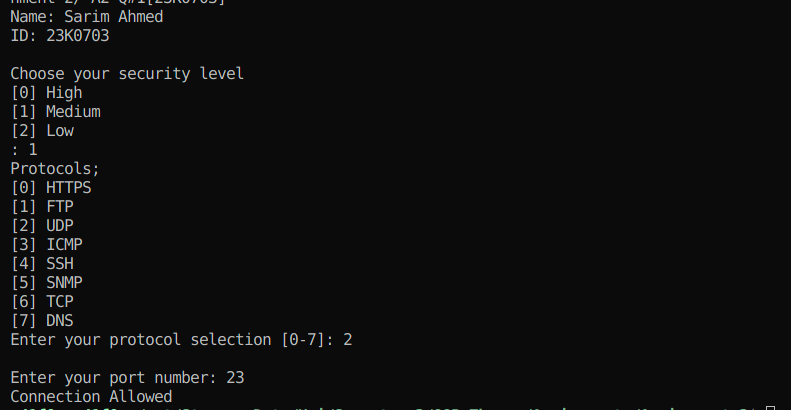
{

cout << "Connection Blocked" << endl;

}

}

Output:



Question 2

Code:

**#**include <iostream>

using namespace std;

class Enemy;

class Player

{

protected:

int playerID, health;

string playerName;

public:

Player(

int playerID,

string playerName)

: playerID(playerID),

playerName(playerName),

health(100) {}

void takeDamage(int damage)

{

health -= damage;

}

int getHealth()

{

return this->health;

}

};

class Weapon

{

string weaponsList[5];

public:

Weapon()

{

weaponsList[0] = "Knife";

weaponsList[1] = "Pistol";

weaponsList[2] = "AK-47";

weaponsList[3] = "RPG";

weaponsList[4] = "Nuke";

}

void use()

{

int option;

// Print all weapons

cout << "==============================\n";

cout << " SELECT YOUR WEAPON!\n";

cout << "==============================\n\n";

cout << "Option\tName\n";

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

{

cout << "[" << i + 1 << "]\t" << weaponsList[i] << endl;

}

// Selecting a weapon

cout << "\nEnter your option: ";

cin >> option;

// Using the weapon

cout << "[+] Your weapon is " << weaponsList[option - 1] << "!" << endl;

}

};

class Character : public Player

{

int level, points;

string experience;

public:

Character(

int playerID,

string playerName)

: level(0),

points(0),

experience("Beginner"),

Player(playerID, playerName) {}

void levelUp()

{

if (experience == "Beginner")

{

experience = "Intermidiate";

}

else if (experience == "Intermidiate")

{

experience = "Advanced";

}

else if (experience == "Advanced")

{

experience = "Expert";

}

}

void playGame(Enemy \*enemy);

};

class Enemy : public Player

{

int damage;

public:

Enemy(int playerID,

string playerName,

int damage)

: Player(playerID, playerName)

{

if (damage > 10)

{

this->damage = 10;

}

else if (damage < 1)

{

this->damage = 1;

}

else

{

this->damage = damage;

}

}

void attack(Character \*character)

{

// Select weapon

Weapon weapons;

weapons.use();

character->takeDamage(this->damage);

}

};

void Character::playGame(Enemy \*enemy)

{

Weapon weapons;

weapons.use();

enemy->takeDamage(5);

this->points += 10;

levelUp();

}

int main()

{

// Solution Header

cout << "Name: Sarim Ahmed" << endl;

cout << "ID: 23K0703" << endl

<< endl;

Enemy enemy(12345, "Baba Bandook", 10);

Character character(12342, "Burqa Avenger");

Enemy \*e = &enemy;

Character \*c = &character;

while (e->getHealth() > 0 && c->getHealth() > 0)

{

cout << "Your health is " << c->getHealth() << endl;

cout << "Enemy's health is " << e->getHealth() << endl;

c->playGame(e);

e->attack(c);

}

if (e->getHealth() <= 0 && c->getHealth() <= 0)

{

cout << "\n\n THE GAME WAS A DRAW." << endl;

}

else if (e->getHealth() <= 0)

{

cout << "\n\n THE CHARACTER WON." << endl;

}

else

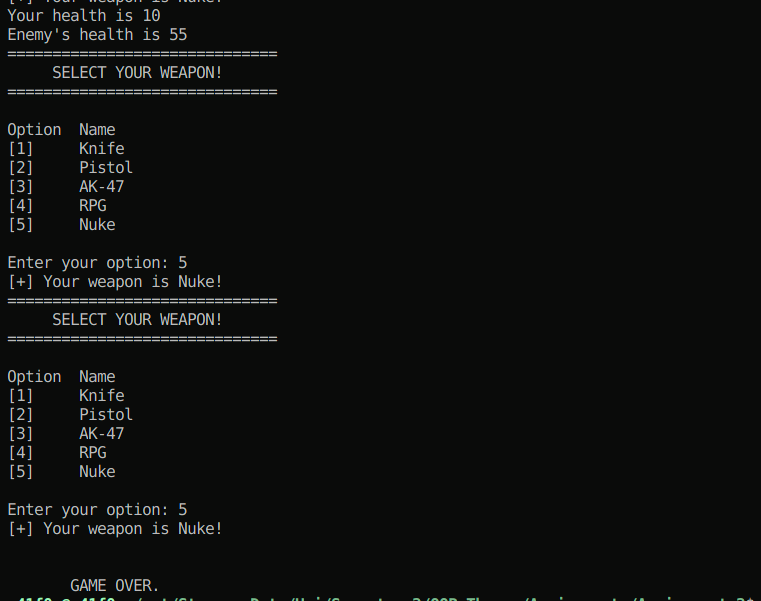
{

cout << "\n\n GAME OVER." << endl;

}

}

Output:



Question 3:

Code:

**#**include <iostream>

using namespace std;

class DarazPersonalData

{

string firstName;

string lastName;

string address;

string city;

string state;

string zip;

string phone;

public:

DarazPersonalData(

string firstName,

string lastName,

string address,

string city,

string state,

string zip,

string phone)

: firstName(firstName),

lastName(lastName),

address(address),

city(city),

state(state),

zip(zip),

phone(phone)

{

}

string getFirstName()

{

return this->firstName;

}

void setFirstName(string firstName)

{

this->firstName = firstName;

}

string getLastName()

{

return this->lastName;

}

void setLastName(string lastName)

{

this->lastName = lastName;

}

string getAddress()

{

return this->address;

}

void setAddress(string address)

{

this->address = address;

}

string getCity()

{

return this->city;

}

void setCity(string city)

{

this->city = city;

}

string getState()

{

return this->state;

}

void setState(string state)

{

this->state = state;

}

string getZip()

{

return this->zip;

}

void setZip(string zip)

{

this->zip = zip;

}

string getPhone()

{

return this->phone;

}

void setPhone(string phone)

{

this->phone = phone;

}

};

class DarazCustomerData : public DarazPersonalData

{

int customerNumber, loyaltyPoints;

public:

DarazCustomerData(

string firstName,

string lastName,

string address,

string city,

string state,

string zip,

string phone,

int customerNumber,

int loyaltyPoints)

: DarazPersonalData(

firstName,

lastName,

address,

city,

state,

zip,

phone)

{

this->customerNumber = customerNumber;

setLoyaltyPoints(loyaltyPoints);

}

int getCustomerNumber()

{

return this->customerNumber;

}

void setCustomerNumber(int customerNumber)

{

this->customerNumber = customerNumber;

}

int getLoyaltyPoints()

{

return this->loyaltyPoints;

}

void setLoyaltyPoints(int loyaltyPoints)

{

if (loyaltyPoints < 0)

{

this->loyaltyPoints = 0;

}

else

{

this->loyaltyPoints = loyaltyPoints;

}

}

};

class DarazLoyaltyProgram

{

public:

void addLoyaltyPoints(DarazCustomerData &customer, int points)

{

int currentPoints = customer.getLoyaltyPoints();

customer.setLoyaltyPoints(currentPoints + points);

}

float redeemLoyaltyPointsForDiscounts(DarazCustomerData &customer, int points)

{

int currentPoints = customer.getLoyaltyPoints();

float discount = 0, discountRate = 3;

if (currentPoints >= points)

{

customer.setLoyaltyPoints(currentPoints - points);

discount = points \* discountRate;

}

cout << "Discount Availed is " << discount << endl;

return discount;

}

void displayTotalLoyaltyPoints(DarazCustomerData &customer)

{

cout << "Loyalty points for " << customer.getFirstName() << " are: " << customer.getLoyaltyPoints() << endl;

}

};

int main()

{

// Solution Header

cout << "Name: Sarim Ahmed" << endl;

cout << "ID: 23K0703" << endl

<< endl;

DarazLoyaltyProgram loyaltyProgram;

DarazCustomerData customerData("Sarim", "Ahmed", "A-214", "Karachi", "Sindh", "101", "03121234567", 1234, 220);

loyaltyProgram.displayTotalLoyaltyPoints(customerData);

loyaltyProgram.redeemLoyaltyPointsForDiscounts(customerData, 200);

loyaltyProgram.displayTotalLoyaltyPoints(customerData);

loyaltyProgram.addLoyaltyPoints(customerData, 100);

loyaltyProgram.displayTotalLoyaltyPoints(customerData);

}

Output:



Question 4:

Code:

**#**include <iostream>

**#**include <functional>

using namespace std;

class User

{

string username, email;

size\_t password;

public:

User(string username, string password, string email)

{

this->username = username;

this->email = email;

this->password = hash(password);

}

bool verifyUser(string email, string password)

{

return this->email == email && this->password == hash(password);

}

size\_t hash(string password)

{

return std::hash<string>{}(password);

}

};

class Post

{

string postId, content, comments[10];

int numComments, likes, views;

public:

Post(string postId, string content) : postId(postId), content(content)

{

numComments = 0;

for (int i = 0; i < 10; i++)

comments[i] = "";

likes = 0;

views = 0;

}

int getViews()

{

return this->views;

}

void promoteViews()

{

this->views \*= 3;

}

int getNumComments()

{

return this->numComments;

}

int getLikes()

{

return this->likes;

}

void promoteLikes()

{

this->likes \*= 2;

}

void comment(string comment)

{

comments[numComments] = comment;

numComments++;

}

void like()

{

likes++;

}

void display()

{

// Add view

views++;

cout << "\nPost ID: " << postId << endl;

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

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

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

cout << "Comments:" << endl;

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

{

cout << "Comment#" << i + 1 << ": " << comments[i] << endl;

}

}

};

class RegularUser : public User

{

int numPostsPosted;

static const int MAX\_FEED\_SIZE = 10;

Post \*feed[MAX\_FEED\_SIZE];

public:

RegularUser(

string username,

string password,

string email)

: User(

username,

password,

email)

{

numPostsPosted = 0;

}

void addToFeed(Post \*post)

{

if (numPostsPosted < 5)

{

numPostsPosted++;

feed[numPostsPosted - 1] = post;

}

}

void viewFeed()

{

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

{

feed[i]->display();

}

}

};

class BusinessUser : public User

{

int numPostsPromoted;

Post \*promotedPosts[10];

public:

BusinessUser(

string username,

string password,

string email)

: User(

username,

password,

email)

{

numPostsPromoted = 0;

}

void promotePost(Post \*post)

{

if (numPostsPromoted < 10)

{

string inputEmail, inputPassword;

cout << "Enter Email: ";

cin >> inputEmail;

cout << "Enter Password: ";

cin >> inputPassword;

if (verifyUser(inputEmail, inputPassword))

{

post->promoteLikes();

post->promoteViews();

promotedPosts[numPostsPromoted] = post;

numPostsPromoted++;

}

}

}

void viewPromotedPosts()

{

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

{

promotedPosts[i]->display();

}

}

};

int main()

{

// Solution Header

cout << "Name: Sarim Ahmed" << endl;

cout << "ID: 23K0703" << endl

<< endl;

Post \*post1 = new Post("12345", "I love getting killed.");

Post \*post2 = new Post("12344", "After life is a high.");

Post \*post3 = new Post("12342", "I love killing people.");

RegularUser user1("uwuMurderer", "12398ru23", "murder@hehe.com");

RegularUser user2("waitingToBeMurdered", "3r4t34t43t", "murdered@haha.com");

BusinessUser user3("murderPromoter786", "1234", "murderIsProductive@business.com");

post1->like();

post1->like();

post1->like();

post1->like();

post2->like();

post2->like();

post2->like();

post3->like();

post3->comment("Great initiative");

user2.addToFeed(post1);

user2.addToFeed(post2);

user2.addToFeed(post3);

user2.viewFeed();

user2.viewFeed();

user3.promotePost(post1);

user3.promotePost(post2);

user3.viewPromotedPosts();

}

Output:

