**ASSIGNMENT#2**

Q#1:

#include<iostream>

using namespace std;

class SecurityTool{

    int **cost**, **numDevices**;

    string **securityLevel**;

    public:

        SecurityTool(string securityLevel, int cost, int numDevices):**numDevices**(numDevices){

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

                this->**securityLevel** *=* securityLevel;

            else{

**cout***<<*"Invalid Security Level"*<<*endl;

                inputSecurityLvl();

            }

            if(cost*>*0)

                this->**cost** *=* cost;

            else{

**cout***<<*"Invalid cost"*<<*endl;

                inputCost();

            }

        }

        void inputSecurityLvl(){

            string **securityLevel**;

            while(**securityLevel** *!=* "Low" *&&* **securityLevel** *!=* "Medium" *&&* **securityLevel** *!=* "High"){

**cout***<<*"Enter security Level again: ";

**cin***>>***securityLevel**;

        }

            this->**securityLevel** *=* **securityLevel**;

        }

        void inputCost(){

            int **cost**;

            while(**cost***<=*0){

**cout***<<*"Enter cost again: ";

**cin***>>***cost**;

            }

            this->**cost** *=* **cost**;

        }

        void performScan(){

**cout***<<*"Performing scan..."*<<*endl;

        }

        string getSecurityLvl(){

            return **securityLevel**;

        }

        void setSecurityLvl(string lvl){

            if(lvl *==* "Low" *||* lvl *==* "Medium" *||* lvl *==* "High")

**securityLevel** *=* lvl;

            else{

**cout***<<*"Invalid Security Level"*<<*endl;

                inputSecurityLvl();

            }

        }

};

class FirewallTool: public SecurityTool{

    int*\** **ports**;

    string *\****protocols**;

    int **numPorts**, **numProtocols**;

    public:

        FirewallTool(string securityLevel, int cost, int numDevices):SecurityTool(securityLevel, cost, numDevices), **numPorts**(0), **numProtocols**(0){}

        void generateList(int num){

            if(num*==*0)

                num *=* 1;

**ports** *=* *new* int[23];

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

**ports**[**i**] *=* *++*num;

**numPorts***++*;

            }

**protocols** *=* *new* **string**[6];

**protocols**[0] *=* "HTTPS";

**protocols**[1] *=* "FTP";

**protocols**[2] *=* "UDP";

**protocols**[3] *=* "ICMP";

**protocols**[4] *=* "SSH";

**protocols**[5] *=* "SNMP";

**numProtocols** *=* 6;

        }

        void performScan(int port, string protocol){

            int **portFound** *=* 0, **protocolFound** *=* 0;

            if(getSecurityLvl() *==* "High"){

                for(int **i***=*0;**i***<***numPorts**;**i***++*)

                    if(**ports**[**i**] *==* port){

**portFound**  *=* 1;

                        break;

                    }

                if(**portFound** *==* 0){

**cout***<<*"Traffic from "*<<*port*<<*" isn't allowed"*<<*endl;

                    return;

                }

                for(int **i***=*0;**i***<***numProtocols**;**i***++*)

                    if(**protocols**[**i**] *==* protocol){

**cout***<<*"Traffic is allowed"*<<*endl;

**protocolFound**  *=* 1;

                        return;

                    }

                if(**protocolFound** *==* 0){

**cout***<<*"Traffic from "*<<*protocol*<<*" isn't allowed"*<<*endl;

                    return;

                }

            }

            else if(getSecurityLvl() *==* "Medium"){

                for(int **i***=*0;**i***<***numPorts**;**i***++*)

                    if(**ports**[**i**] *==* port){

**portFound**  *=* 1;

                        break;

                    }

                if(**portFound** *==* 0){

                    if(**ports**[**numPorts***-*1]*+*1 *==* port *||* **ports**[**numPorts***-*1]*+*2 *==* port)

**portFound** *=* 1;

                    else{

**cout***<<*"Traffic from "*<<*port*<<*" isn't allowed"*<<*endl;

                        return;

                    }

                }

                for(int **i***=*0;**i***<***numProtocols**;**i***++*)

                    if(**protocols**[**i**] *==* protocol){

**cout***<<*"Traffic is allowed"*<<*endl;

**protocolFound**  *=* 1;

                        return;

                    }

                if(**protocolFound** *==* 0){

**cout***<<*"Traffic from "*<<*protocol*<<*" isn't allowed"*<<*endl;

                    return;

                }

            }

            else if(getSecurityLvl()*==*"Low"){

                for(int **i***=*0;**i***<***numPorts**;**i***++*)

                    if(**ports**[**i**] *==* port){

**portFound**  *=* 1;

                        break;

                    }

                if(**portFound** *==* 0){

                    if(**ports**[**numPorts***-*1] *<* port *&&* **ports**[**numPorts***-*1]*+*5 *>=* port)

**portFound** *=* 1;

                    else{

**cout***<<*"Traffic from "*<<*port*<<*" isn't allowed"*<<*endl;

                        return;

                    }

                }

                for(int **i***=*0;**i***<***numProtocols**;**i***++*)

                    if(**protocols**[**i**] *==* protocol){

**cout***<<*"Traffic is allowed"*<<*endl;

**protocolFound**  *=* 1;

                        return;

                    }

                if(**protocolFound** *==* 0){

                    if(protocol *==* "TCP" *||* protocol *==* "DNS")

**cout***<<*"Traffic is allowed"*<<*endl;

                    else

**cout***<<*"Traffic from "*<<*protocol*<<*" isn't allowed"*<<*endl;

                    return;

                }

            }

        }

    ~FirewallTool(){

*delete[]* **ports**;

*delete[]* **protocols**;

    }

};

int main() {

**cout***<<*endl*<<*endl*<<*"Coded By: Sahil Latif (23i0763)"*<<*endl*<<*endl;

    FirewallTool **f1**("High", 1000, 5);

**f1**.generateList(7);

**f1**.performScan(23, "HTTPS");

**f1**.performScan(33, "HTTPS");

**f1**.performScan(13, "DNS");

**f1**.setSecurityLvl("Medium");

**f1**.performScan(15, "DNS");

**f1**.performScan(23, "ICMP");

**f1**.performScan(32, "SSH");

**f1**.setSecurityLvl("Low");

**f1**.performScan(34, "UDP");

**f1**.performScan(36, "UDP");

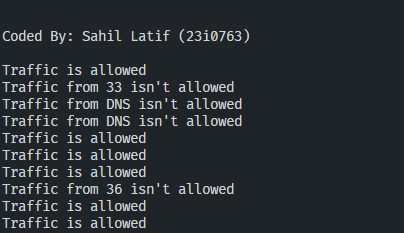
**f1**.performScan(23, "SNMP");

**f1**.performScan(23, "DNS");

    return 0;

}

Output:



Q#2:

#include<iostream>

using namespace std;

class Player{

    protected:

        int **playerID**, **health**;

        string **playerName**;

    public:

        Player(int ID, string name):**playerID**(ID),**playerName**(name),**health**(100){}

        void decreaseHealth(int n){

            if(**health***>*n)

**health***-=*n;

            else

**health** *=* 0;

        }

        string getName(){

            return **playerName**;

        }

        void display(){

**cout***<<*"Player ID: "*<<***playerID***<<*endl;

**cout***<<*"Player Name: "*<<***playerName***<<*endl;

**cout***<<*"Player Health: "*<<***health***<<*endl*<<*endl;

        }

};

class Weapon{

    string *\****weaponList**;

    int **numWeapons**;

    public:

        Weapon(string weapons[], int nWeapons){

            if(nWeapons*>=*5){

**weaponList** *=* *new* **string**[nWeapons];

                for(int **i***=*0;**i***<*nWeapons;**i***++*)

**weaponList**[**i**] *=* **weapons**[**i**];

**numWeapons** *=* nWeapons;

            }

            else

**cout***<<*"Can't set weapon list, need at least 5 weapons to add"*<<*endl;

        }

        string use(){

            for(int **i***=*0;**i***<***numWeapons**;**i***++*)

**cout***<<***i***+*1*<<*". "*<<***weaponList**[**i**]*<<*endl;

            int **weaponInd**;

            do{

**cout***<<*"Choose weapon that you want to use: ";

**cin***>>***weaponInd**;

            }while(**weaponInd***<*0 *||* **weaponInd***>***numWeapons**);

            return **weaponList**[**weaponInd***-*1];

        }

        ~Weapon(){

*delete[]* **weaponList**;

        }

};

class Enemy;

class Character:public Player, public Weapon{

    int **level**, **points**;

    string **experience**;

    static string **allExperiences**[4];

public:

    Character(int ID, string name, string weapons[],int nWeapon, int lvl*=*0, int points*=*0, string exp*=*"Beginner"):Player(ID, name),Weapon(weapons, nWeapon),**level**(lvl),**points**(points),**experience**(exp){}

    void levelUp(){

**points***+=*10;

**level***++*;

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

            if(**experience***==***allExperiences**[**i**]){

**experience***=***allExperiences**[**i***+*1];

                break;

            }

    }

    void playGame(Enemy &enemy);

    void display(){

**cout***<<*"Character ID: "*<<***playerID***<<*endl;

**cout***<<*"Character Name: "*<<***playerName***<<*endl;

**cout***<<*"Character Health: "*<<***health***<<*endl;

**cout***<<*"Character Level: "*<<***level***<<*endl;

**cout***<<*"Character Points: "*<<***points***<<*endl;

**cout***<<*"Character Experience: "*<<***experience***<<*endl*<<*endl;

    }

};

class Enemy:public Player, public Weapon{

    int **damage**;

    public:

        Enemy(int ID, string name, string weapons[],int nWeapon,int damage):Player(ID,name),Weapon(weapons,nWeapon){

            if(damage*>=*0 *&&* damage*<=*10)

                this->**damage** *=* damage;

            else

                damage *=* 5;

        }

        void attack(Character &ch){

**cout***<<*"Enemy "*<<***playerName***<<*" choosing weapon"*<<*endl;

            string **weapon** *=* use();

**cout***<<***playerName***<<*" using " *<<***weapon***<<*endl*<<*endl;

**ch**.decreaseHealth(**damage**);

        }

        void display(){

**cout***<<*"Enemy ID: "*<<***playerID***<<*endl;

**cout***<<*"Enemy Name: "*<<***playerName***<<*endl;

**cout***<<*"Enemy Health: "*<<***health***<<*endl;

**cout***<<*"Enemy Damage: "*<<***damage***<<*endl*<<*endl;

        }

};

void Character::playGame(Enemy &enemy){

**cout***<<*"Character "*<<***playerName***<<*" playing game with enemy "*<<***enemy**.getName()*<<*endl;

**cout***<<*"Character choosing weapon: "*<<*endl;

    string **weapon** *=* use();

**cout***<<*getName()*<<*" using " *<<***weapon***<<*endl*<<*endl;

    levelUp();

**enemy**.decreaseHealth(5);

}

string Character::**allExperiences**[] *=* {"Beginner", "Intermediate", "Advanced", "Expert"};

int main() {

**cout***<<*endl*<<*endl*<<*"Coded By: Sahil Latif (23i0763)"*<<*endl*<<*endl;

    string **w**[6] *=* {"Sword", "Gun", "Bow", "Knife", "Sniper", "Hammer"};

    Character **Hero**(12, "Superman",**w**,6);

    Enemy **En**(10, "V1", **w**,6,8);

**Hero**.display();

**En**.display();

**Hero**.playGame(**En**);

**En**.attack(**Hero**);

**Hero**.display();

**En**.display();

**Hero**.playGame(**En**);

**Hero**.display();

**En**.display();

**Hero**.playGame(**En**);

**Hero**.display();

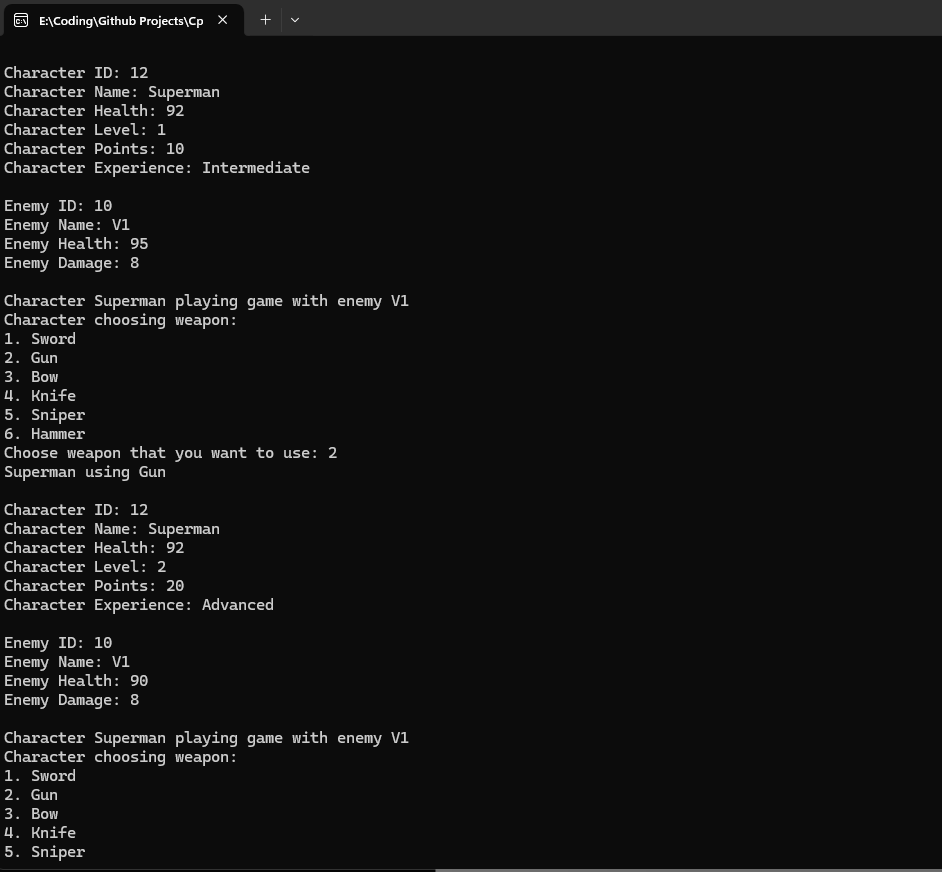
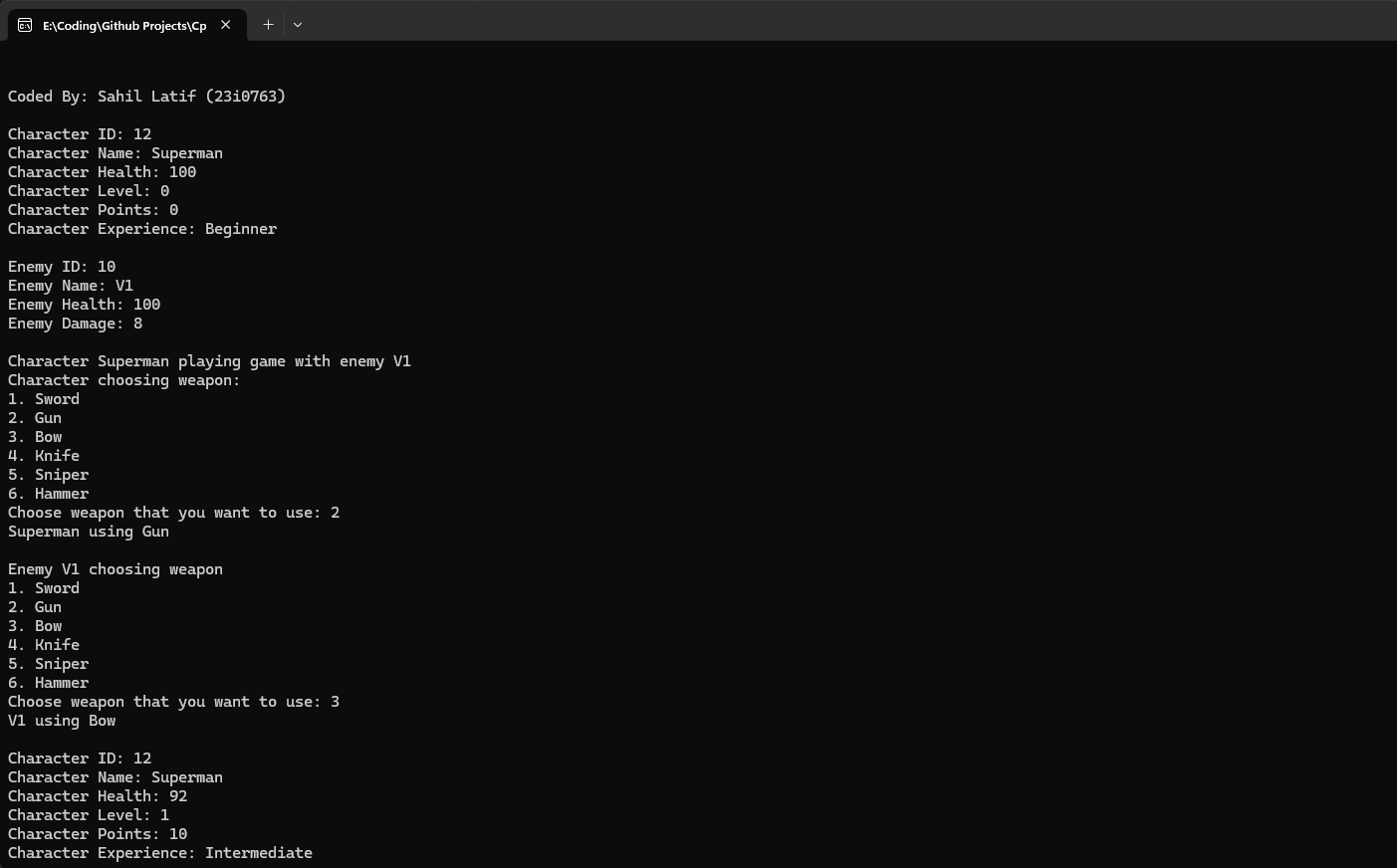
**En**.display();

**Hero**.playGame(**En**);

**Hero**.display();

**En**.display();

}

Output:

Q3:

#include<iostream>

using namespace std;

class DarazPersonData{

    string **firstName**, **lastName**, **address**, **city**, **state**, **zip**, **phone**;

    public:

        DarazPersonData(string fN, string lN, string add, string city, string state, string zip, string phone):**firstName**(fN),**lastName**(lN),**address**(add),**city**(city), **state**(state), **zip**(zip), **phone**(phone){}

        string getFirstName(){return **firstName**;}

        string getLastName(){return **lastName**;}

        string getAddress(){return **address**;}

        string getCity(){return **city**;}

        string getState(){return **state**;}

        string getZip(){return **zip**;}

        string getPhone(){return **phone**;}

        void setFirstName(string fN){**firstName** *=* fN;}

        void setLasttName(string lN){**lastName** *=* lN;}

        void setAddress(string add){**address** *=* add;}

        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;}

};

class DarazCustomerData{

    static int **totalCustomers**;

    const int **customerNumber**;

    int **loyaltyPoints**;

    public:

        DarazCustomerData(int loyaltyPoints):**customerNumber**(*++***totalCustomers**){

            if(loyaltyPoints*<*0){

**cout***<<*"invalid loyalty points!"*<<*endl;

                this->**loyaltyPoints** *=* 0;

            }

            else

                this->**loyaltyPoints** *=* loyaltyPoints;

        }

        int getCustomerNumber(){return **customerNumber**;}

        int getLoyaltyPoints(){return **loyaltyPoints**;}

        void setLoyaltyPoints(int points){**loyaltyPoints** *=* points;}

};

class DarazLoyaltyProgram:public DarazPersonData, public DarazCustomerData{

    public:

        DarazLoyaltyProgram(string fN, string lN, string add, string city, string state, string zip, string phone, int points):DarazPersonData(fN,lN,add,city,state, zip, phone), DarazCustomerData(points){}

        void addLoyaltyPoints(int points){

            if(points*>*0){

                setLoyaltyPoints(getLoyaltyPoints()*+*points);

**cout***<<*points*<<*" Points added"*<<*endl;

            }

            else

**cout***<<*"Negative Points can't be added"*<<*endl;

        }

        void redeemLoyaltyPoints(int points){

            int **cPoints** *=* getLoyaltyPoints();

            if(points*<=***cPoints**){

                setLoyaltyPoints(**cPoints** *-* points);

**cout***<<*"discount redeemed for "*<<*points*<<*" Points"*<<*endl;

            }

            else

**cout***<<*"Not enough loyalty Points to redeem discount"*<<*endl;

        }

        void displayLoyaltyPoints(){

**cout***<<*"Customer Name: "*<<*getFirstName()*<<*" "*<<*getLastName()*<<*endl;

**cout***<<*"Loyalty Points: "*<<*getLoyaltyPoints()*<<*endl;

        }

};

int DarazCustomerData::**totalCustomers** *=* 0;

int main() {

**cout***<<*endl*<<*endl*<<*"Coded By: Sahil Latif (23i0763)"*<<*endl*<<*endl;

    DarazLoyaltyProgram **p1**("Sahil", "Latif", "Shah Latif", "Karachi", "Town", "72727", "03123456789",10);

**p1**.displayLoyaltyPoints();

**p1**.addLoyaltyPoints(20);

**p1**.displayLoyaltyPoints();

**p1**.redeemLoyaltyPoints(20);

**p1**.displayLoyaltyPoints();

**p1**.redeemLoyaltyPoints(15);

**p1**.displayLoyaltyPoints();

**p1**.addLoyaltyPoints(*-*20);

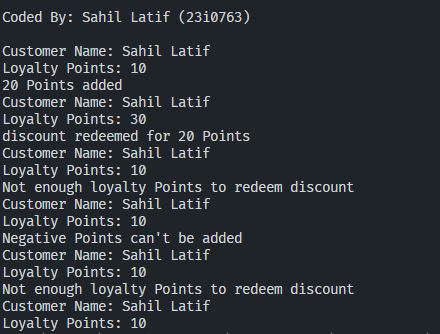
**p1**.displayLoyaltyPoints();

**p1**.redeemLoyaltyPoints(15);

**p1**.displayLoyaltyPoints();

}

Output:



Q4:

#include<iostream>

#include<cmath>

using namespace std;

class User{

    string **username**, **email**, **password**;

    public:

        User(string username, string email, string password):**username**(username), **email**(email){

            this->**password** *=* encode(password);

        }

        string encode(string pass){

            string **hashedPass**, **salt** *=* "iamsaltusedforencoding";

**pass**.append(**salt**);

            int **len** *=* **pass**.length();

            for(int **i***=*0;**i***<***len**;**i***++*){

                for(int **j***=*1;**j***<*4;**j***++*){

                    char **c** *=* char (**pass***[***i***]+*pow(**len***%*3*+*2,**j**));

**hashedPass**.push\_back(**c**);

                }

            }

**hashedPass***+=*to\_string(**len***%*3*+*2);

            return **hashedPass**;

        }

        string decode(string hash){

            string **pass**, **salt** *=* "iamsaltusedforencoding";

            for(int **i***=*0;**i***<***hash**.length();**i***+=*3){

                char **c** *=* char (**hash***[***i***]-* (**hash***[***hash**.length()*-*1*]*)) *+* '0';

**pass**.push\_back(**c**);

            }

**pass**.resize(**pass**.length()*-***salt**.length()*-*1);

            return **pass**;

        }

        string getUsername(){return **username**;}

};

class Post{

    int **postID**, **likes**,**views**, **numComments**;

    string **content**, *\****comments**;

    static int **totalPosts**;

    public:

        Post(string content):**postID**(*++***totalPosts**), **content**(content), **likes**(0),**views**(0), **numComments**(0){}

        void addComment(string comment){

            string *\****newCom** *=* *new* **string**[**numComments***+*1];

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

**newCom**[**i**] *=* **comments**[**i**];

**newCom**[**numComments***++*] *=* comment;

**comments** *=* **newCom**;

**views***++*;

        }

        int getLikes(){return **likes**;}

        int getViews(){return **views**;}

        int getNumComments(){return **numComments**;}

        void setLikes(int likes){this->**likes** *=* likes;}

        void setViews(int views){this->**views** *=* views;}

        void setNumComments(int numComments){this->**numComments** *=* numComments;}

        void likePost(){**likes***++*;**views***++*;}

        void displayDetails(){

**cout***<<*"PostID: "*<<***postID***<<*endl;

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

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

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

**cout***<<*"Comments: "*<<***numComments***<<*endl;

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

**cout***<<***i***+*1*<<*". "*<<***comments**[**i**]*<<*endl;

**cout***<<*endl;

**views***++*;

        }

};

int Post::**totalPosts** *=* 0;

class RegularUser:public User{

    protected:

        static int **maxFeed**;

        Post *\*\****myposts**;

        Post *\*\****myfeed**;

        int **numPosts**, **numFeed**;

    public:

        RegularUser(string username, string email, string password):User(username, email, password),**numPosts**(0),**numFeed**(0){}

        void addPost(Post \*post){

            if(**numPosts***<*5){

                Post *\*\****newPosts** *=* *new* Post*\**[**numPosts***+*1];

                for(int **i***=*0;**i***<***numPosts**;**i***++*)

**newPosts**[**i**] *=* **myposts**[**i**];

**newPosts**[**numPosts***++*] *=* post;

**myposts** *=* **newPosts**;

                addPostToFeed(post);

            }

            else

**cout***<<*"Max Posts limit reached"*<<*endl;

        }

        void addPostToFeed(Post \*post){

            if(**numFeed***<***maxFeed**){

                Post *\*\****newPosts** *=* *new* Post*\**[**numFeed***+*1];

                for(int **i***=*0;**i***<***numFeed**;**i***++*)

**newPosts**[**i**] *=* **myfeed**[**i**];

**newPosts**[**numFeed***++*] *=* post;

**myfeed** *=* **newPosts**;

            }

            else

**cout***<<*"Max Feed limit reached"*<<*endl;

        }

        void viewMyPosts(){

**cout***<<*endl*<<*"Username: "*<<*getUsername()*<<*endl*<<*endl;

            for(int **i***=*0;**i***<***numPosts**;**i***++*)

**myposts**[**i**]->displayDetails();

        }

        void viewFeed(){

            for(int **i***=*0;**i***<***numFeed**;**i***++*)

**myfeed**[**i**]->displayDetails();

        }

};

int RegularUser::**maxFeed** *=* 10;

class BusinessUser: public RegularUser{

    int **numPromotedPosts**;

    public:

        BusinessUser(string username, string email, string password):RegularUser(username, email, password), **numPromotedPosts**(0){}

        void promotePost(Post &post){

            if(**numPromotedPosts***<*10){

**post**.setLikes(**post**.getLikes()*\**2);

**post**.setViews(**post**.getViews()*\**3);

**numPromotedPosts***++*;

            }

            else

**cout***<<*"Max Promoted Posts limit reached"*<<*endl;

        }

        void trackLikes(Post post){

**cout***<<*"Post Likes: "*<<***post**.getLikes()*<<*endl;

        }

        void trackComments(Post post){

**cout***<<*"Post Comments: "*<<***post**.getNumComments()*<<*endl;

        }

        void trackView(Post post){

**cout***<<*"Post Views: "*<<***post**.getViews()*<<*endl;

        }

};

int main() {

**cout***<<*endl*<<*endl*<<*"Coded By: Sahil Latif (23i0763)"*<<*endl*<<*endl;

    Post **posts**[10] *=* {Post("Post 1"), Post("Post 2"), Post("Post 3"), Post("Post 4"), Post("Post 5"), Post("Post 6"), Post("Post 7"), Post("Post 8"), Post("Post 9"), Post("Post 10")};

    RegularUser **u1**("Sahil", "sahillatif@gmail.com", "Sahil123");

    BusinessUser **u2**("Ali", "alikhan786@yahoo.com","Ali123");

**u1**.addPost(*&***posts**[0]);

**u1**.viewMyPosts();

**u1**.addPost(*&***posts**[2]);

**u1**.addPost(*&***posts**[3]);

**u1**.addPost(*&***posts**[4]);

**u1**.addPost(*&***posts**[5]);

**u1**.addPost(*&***posts**[8]);

**u2**.addPost(*&***posts**[1]);

**u2**.addPost(*&***posts**[6]);

**u2**.addPost(*&***posts**[7]);

**posts**[1].addComment("Nice Post");

**posts**[1].addComment("Great Post");

**posts**[1].likePost();

**posts**[2].likePost();

**posts**[3].likePost();

**u1**.addPostToFeed(*&***posts**[1]);

**u1**.addPostToFeed(*&***posts**[6]);

**cout***<<*endl*<<*endl*<<*"Before Promoting Post"*<<*endl*<<*endl;

**u2**.trackLikes(**posts**[1]);

**u2**.trackComments(**posts**[1]);

**u2**.trackView(**posts**[1]);

**u2**.promotePost(**posts**[1]);

**cout***<<*endl*<<*endl*<<*"After Promoting Post"*<<*endl*<<*endl;

**u2**.trackLikes(**posts**[1]);

**u2**.trackComments(**posts**[1]);

**u2**.trackView(**posts**[1]);

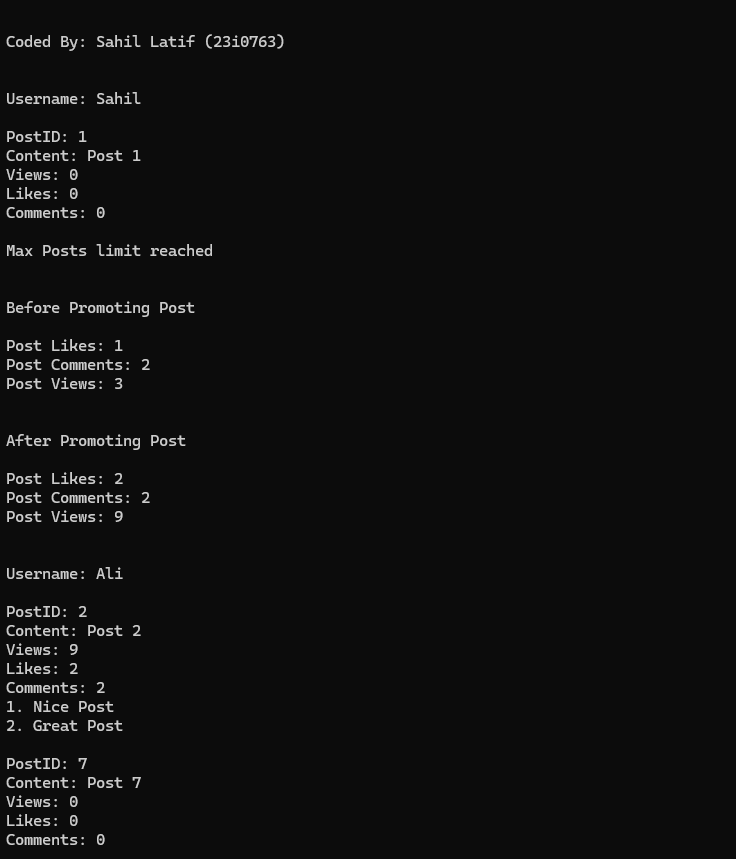
**cout***<<*endl;

**u2**.viewMyPosts();

**u1**.viewFeed();

}

Output:



A screenshot of a computer

Description automatically generated