**Assignment 4**

Q1.

CODE:

//friend class in cpp

#include<iostream>

#include<string>

using namespace std;

#define an account\_list::no\_of\_acc

class account{

int acc\_no;

int balance;

public:

friend class account\_list;

};

class account\_list{

static int no\_of\_acc;

account list[100];

public:

void add\_acc();

void withdraw(int);

void deposit(int);

void display\_acc(int);

void const disp\_count(){

cout<<"Number of Active Accounts: "<<an<<"\n\n";

}

};

int account\_list::no\_of\_acc=0;

void account\_list::add\_acc(){

if(an>100){

cout<<"ACCOUNT LIMIT REACHED. PLEASE CONTACT THE MANAGER.\ n";

return;

}

++an;

list[an].acc\_no=an;

cout<<"Enter Account Balance: ";

cin>>list[an].balance;

cout<<"Alloted Account Number: A00"<<list[an].acc\_no<<"\n";

}

void account\_list::withdraw(int a){

if(a<=0 || a>an){

cout<<"Invalid Account Number!\n";

return;

}

cout<<"Available Balance in account A00"<<a<<": Rs. "<<list[a].balance<<"\n";

int amt;

cout<<"Enter Amount to be Withdrawn: ";

cin>>amt;

if(amt>list[a].balance){

cout<<"Withdraw Amount Exceeded!\n";

return;

}

list[a].balance = list[a].balance - amt;

cout<<"Account Updated!\n";

}

void account\_list::deposit(int a){

if(a<=0 || a>an){

cout<<"Invalid Account Number!\n";

return;

}

cout<<"Available Balance in account A00"<<a<<": Rs. "<<list[a].balance<<"\n";

int amt;

cout<<"Enter Amount to be Deposit: ";

cin>>amt;

list[a].balance = list[a].balance + amt;

cout<<"Account Updated!\n";

}

void account\_list::display\_acc(int a){

if(a<=0 || a>an){

cout<<"Invalid Account Number!\n";

return;

}

cout<<"Account Number: A00"<<list[a].acc\_no<<"\n";

cout<<"Account Balance: "<<list[a].balance<<"\n";

}

int main(){

account\_list ob;

int ch;

string acc;

do{

cout<<"\n\*\*\*BANK TRANSACTION SYSTEM\*\*\*\n\n";

cout<<"1. ADD ACCOUNT.\n";

cout<<"2. WITHDRAW AMOUNT.\n";

cout<<"3. DEPOSIT AMOUNT.\n";

cout<<"4. DISPLAY ACCOUNT.\n";

cout<<"5. DISPLAY NUMBER OF ACTIVE ACCOUNTS.\n";

cout<<"6. EXIT.\n";

cout<<"Enter Your Choice: ";

cin>>ch;

switch(ch){

case 1:

ob.add\_acc();

break;

case 2:

cout<<"\nEnter Acoount Number(format A00X): ";

cin>>acc;

ob.withdraw(acc[3] - '0');

break;

case 3:

cout<<"\nEnter Acoount Number(format A00X): ";

cin>>acc;

ob.deposit(acc[3] - '0');

break;

case 4:

cout<<"\nEnter Acoount Number(format A00X): ";

cin>>acc;

ob.display\_acc(acc[3] - '0');

break;

case 5:

ob.disp\_count();

break;

case 6:

exit(0);

break;

default:

cout<<"INVALID CHOICE!!!!\n\n";

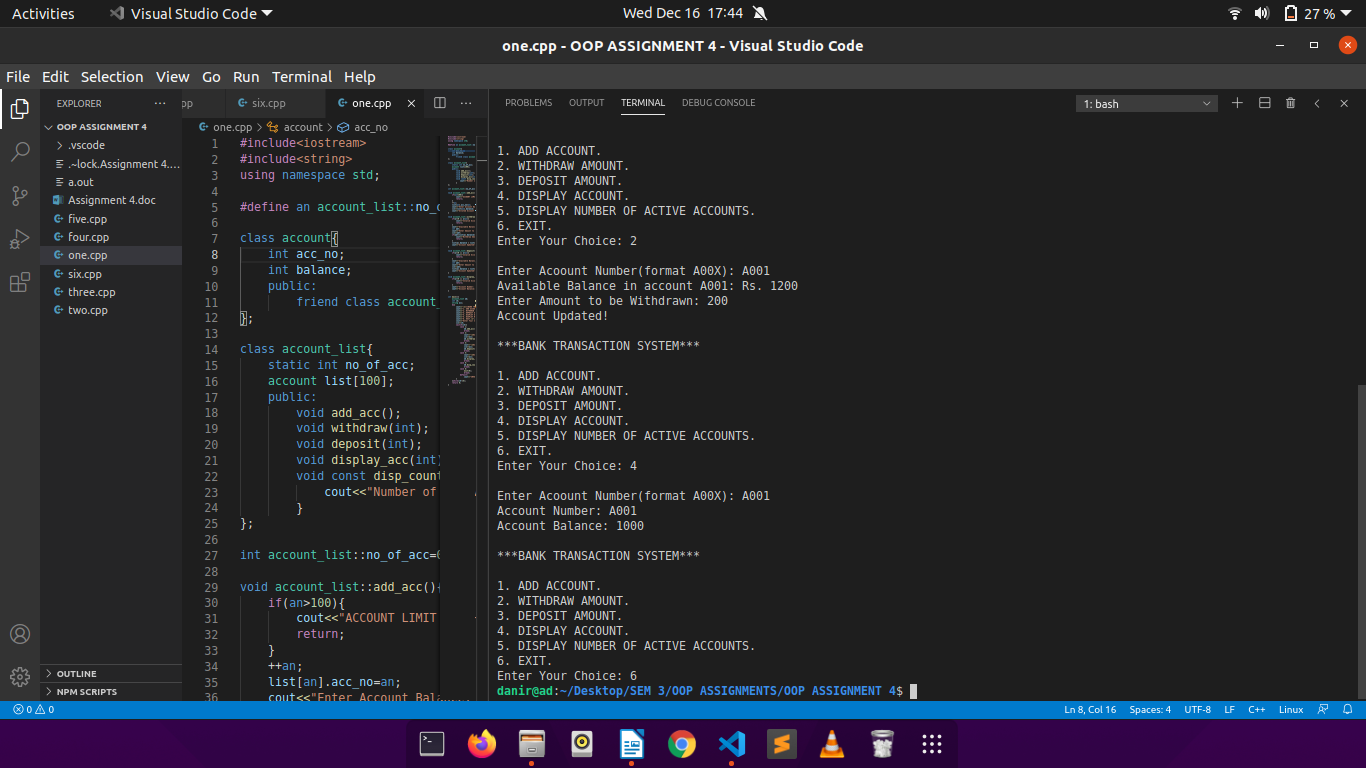
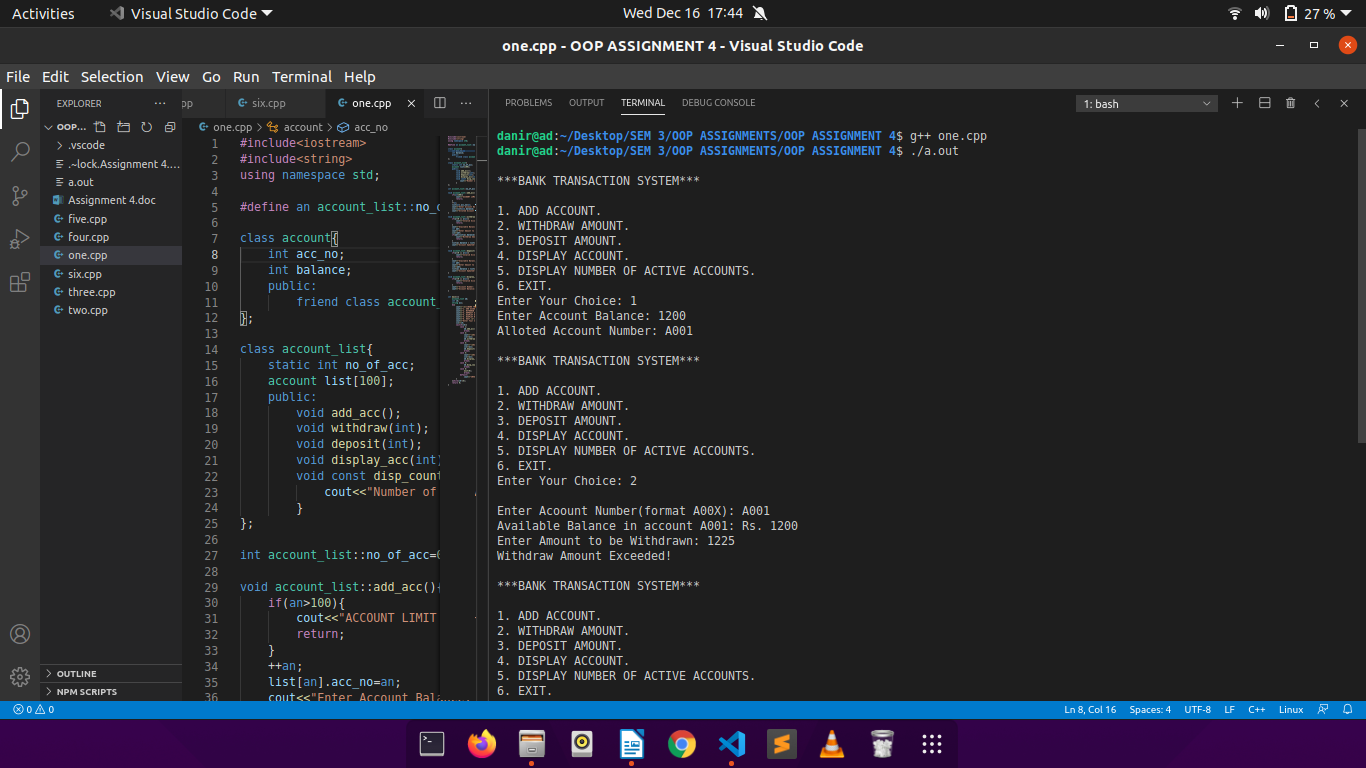
}

}while(ch!=6);

return 0;

}

OUTPUT:



Q2.

CODE:

//operator overloading in cpp(1)

#include<iostream>

using namespace std;

class complex{

int real;

int img;

public:

complex(int a=0, int b=0){

real=a;

img=b;

}

complex operator+(complex c){

complex t;

t.real = real + c.real;

t.img = img + c.img;

return t;

}

complex operator-(complex c){

complex t;

t.real = real - c.real;

t.img = img - c.img;

return t;

}

friend ostream& operator<<(ostream &o, complex c){

o<<"The resultant complex number is: "<<c.real<<" + "<<c.img<<"i\n";

return o;

}

friend istream& operator>>(istream &i, complex &c){

cout<<"Real: ";

i>>c.real;

cout<<"Imag: ";

i>>c.img;

return i;

}

};

int main(){

int a,b;

complex c1,c2;

cout<<"Enter complex numbers:\n";

cin>>c1;

cin>>c2;

complex c3;

c3=c1+c2;

cout<<c3;

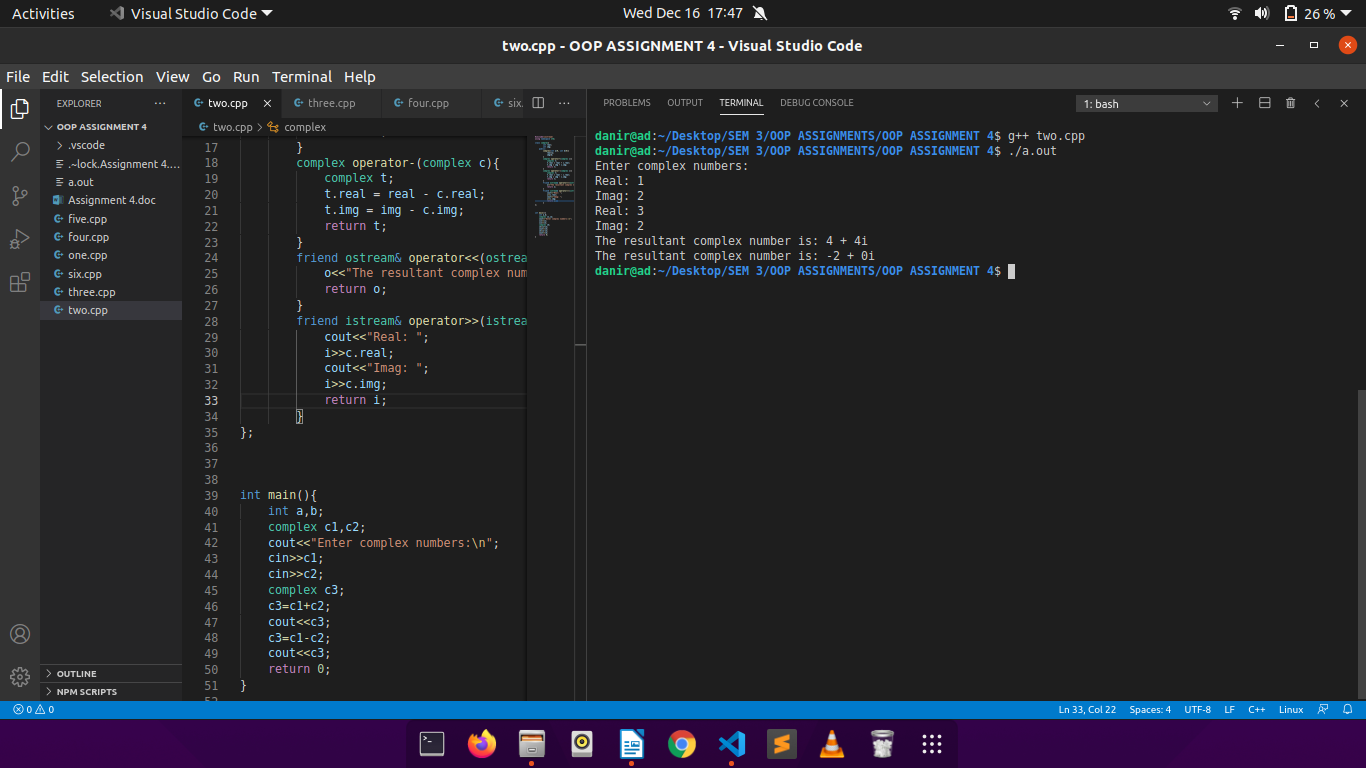
c3=c1-c2;

cout<<c3;

return 0;

}

OUTPUT:



Q3.

CODE:

//operator overloading in cpp(2)

#include <iostream>

#include <assert.h>

using namespace std;

class Array {

int size;

int\* arr;

public:

int& operator[](int i)

{

if (i < 0 || i >= size) {

cout << "Index out of bounds.\n";

return arr[0];

}

return arr[i];

}

Array(int sz = 0, int val = 0)

{

size = max(0, sz);

if (size == 0) {

arr = NULL;

}

else {

arr = new int[size];

for (int i = 0; i < size; ++i) {

arr[i] = val;

}

}

}

Array(Array& temp)

{

size = temp.size;

if (size == 0) {

arr = NULL;

}

else {

arr = new int[size];

for (int i = 0; i < size; ++i) {

arr[i] = temp[i];

}

}

}

void operator=(const Array& a)

{

delete[] arr;

size = a.size;

arr = new int[size];

for (int i = 0; i < size; ++i) {

arr[i] = a.arr[i];

}

}

Array(int \*temp, int sz)

{

size = sz;

if (size == 0) {

arr = NULL;

}

else {

arr = new int[size];

for (int i = 0; i < size; ++i) {

arr[i] = temp[i];

}

}

}

friend Array operator+(Array a, Array b)

{

assert(a.size == b.size);

Array c(a.size);

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

c[i] = a[i] + b[i];

}

return c;

}

friend Array operator\*(Array a, int n)

{

Array b(a);

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

b[i] = a[i] \* n;

}

return b;

}

friend ostream& operator<<(ostream& out, Array a)

{

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

out << a[i] << " ";

}

out << "\n";

return out;

}

~Array()

{

delete[] arr;

}

};

int main() {

Array a(5, 5);

Array b = \*(new Array(5, 7));

int c[] = {1, 2, 3, 4, 5};

Array d(c, (sizeof(c) / sizeof(int)));

Array e;

e = b + d;

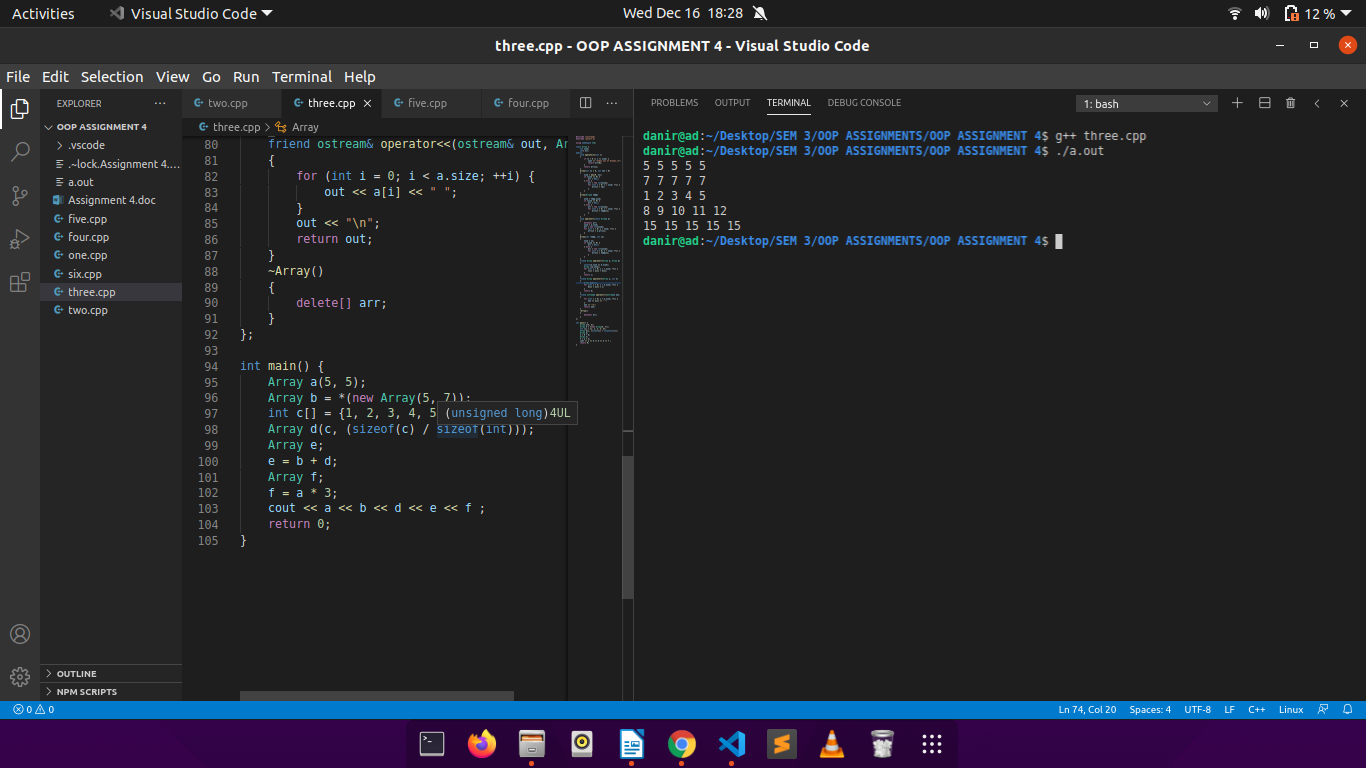
Array f;

f = a \* 3;

cout << a << b << d << e << f ;

return 0;

}

OUTPUT:

Q4.

CODE:

//operator overloading in cpp(3)

#include <iostream>

#include <string.h>

using namespace std;

class String {

int len;

char \*s;

public:

String()

{

len = 0;

s = new char[len + 1];

strcpy(s, "");

}

String(char\* temp)

{

len = 0;

while (temp[len] != '\0')

++len;

s = new char[len + 1];

char\* ss = new char[len + 1];

strcpy(ss, temp);

strcpy(s, ss);

delete[] ss;

}

String(String& a)

{

len = a.len;

s = new char[len + 1];

char\* ss = new char[len + 1];

strcpy(ss, a.s);

strcpy(s, ss);

delete[] ss;

}

void operator=(const String& a)

{

s = a.s;

len = a.len;

}

friend String operator+(String& a, String& b)

{

char\* temp;

temp = new char[a.len + b.len + 1];

int i = 0;

while (i < a.len) {

temp[i] = a.s[i];

++i;

}

int j = 0;

while (j < b.len) {

temp[i] = b.s[j];

++i;

++j;

}

temp[i] = '\0';

String c(temp);

delete[] temp;

return c;

}

friend bool operator>(String& a, String& b)

{

if (strcmp(a.s, b.s) > 0)

return true;

return false;

}

friend bool operator<(String& a, String& b)

{

if (strcmp(a.s, b.s) < 0)

return true;

return false;

}

friend bool operator==(String& a, String& b)

{

if (strcmp(a.s, b.s) == 0)

return true;

return false;

}

friend ostream& operator<<(ostream& out, String& a)

{

out << a.s;

return out;

}

// ~String()

// {

// delete[] s;

// }

};

int main() {

char s1[] = "hello ", s2[] = "world";

String a(s1);

String b(s2);

String c;

c = a + b;

cout << c << "\n";

String d = b;

if (d == b) {

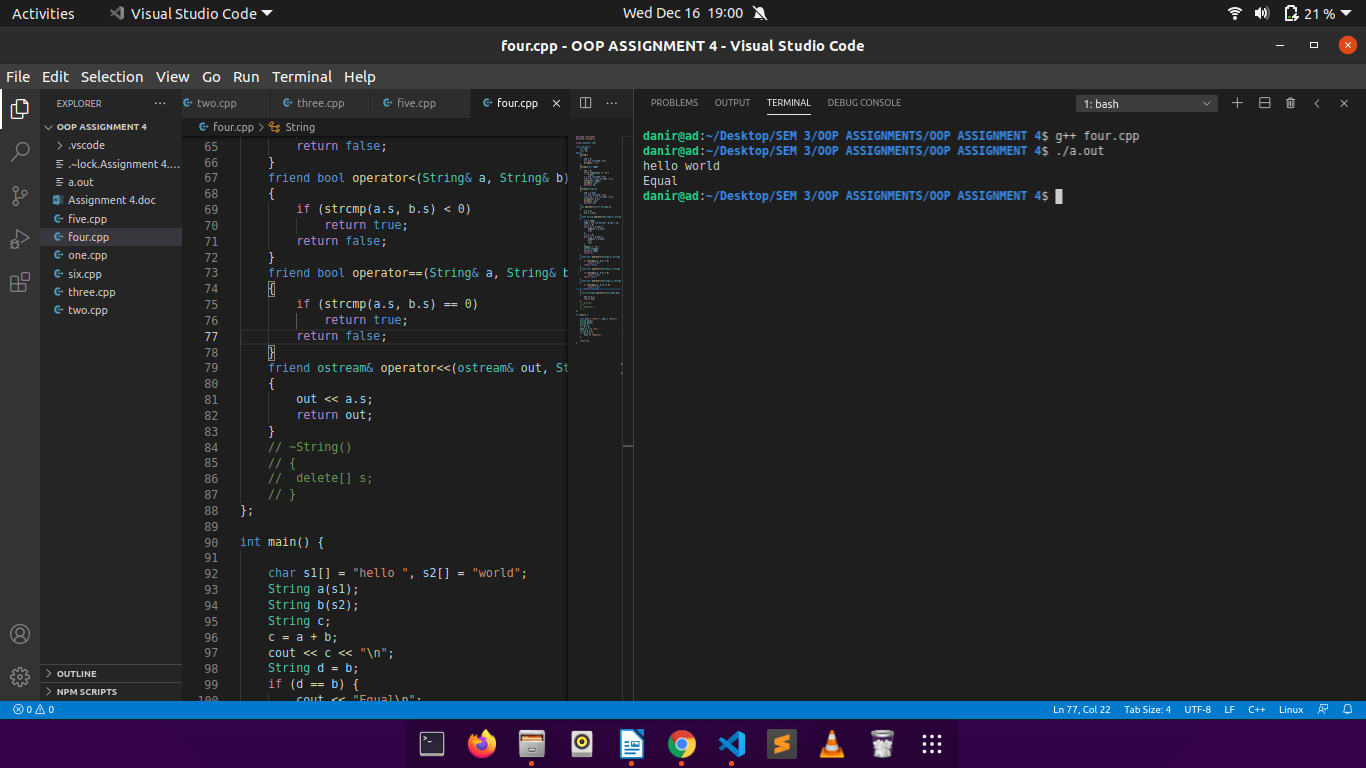
cout << "Equal\n";

}

return 0;

}

OUTPUT:



Q5.

CODE:

//operator overloading and copy constructors in cpp

#include<iostream>

#include<string.h>

using namespace std;

class String{

char\* c;

int size;

int \*p;

public:

String(int s=0){

size=s;

if(s==0)

c=NULL;

else{

c=new char[size];

c[0]='\0';

}

p=new int;

\*p=1;

}

String(const String& s){

size=s.size;

c=s.c;

p=s.p;

(\*p)++;

}

void operator=(String s){

if(c!=NULL)

(\*p)--;

if((\*p)==0){

delete[] c;

delete p;

}

c=s.c;

size=s.size;

p=s.p;

(\*p)++;

}

~String(){

(\*p)--;

if((\*p)==0){

delete p;

delete[] c;

}

}

void const showInstanceCount(){

cout<<"ACTIVE: "<<(\*p)<<endl;

}

};

int main(){

String s1(10);

String s2(s1);

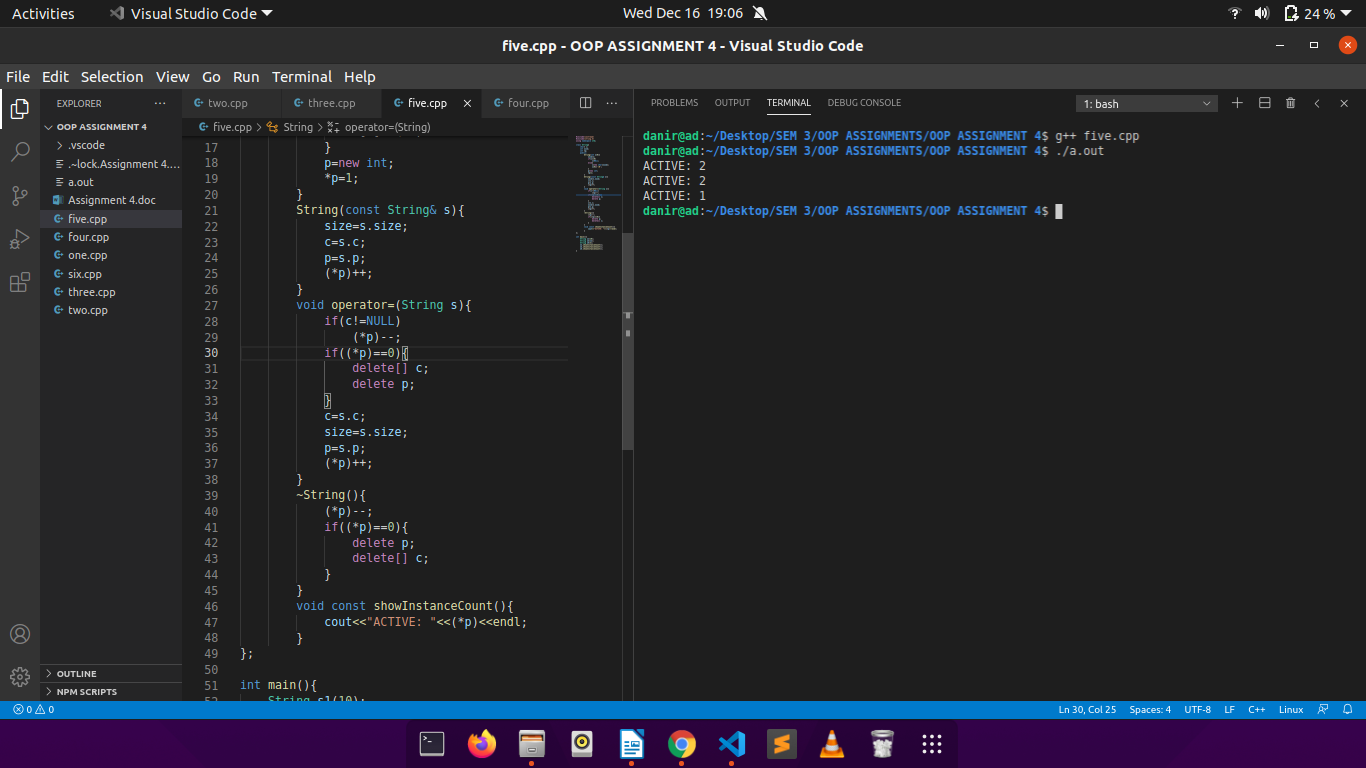
String s4(3);

s1.showInstanceCount();

s2.showInstanceCount();

s4.showInstanceCount();

}

OUTPUT:

Q6.

CODE:

//operator overloading in cpp(4)

#include<iostream>

using namespace std;

#define nstu student\_list::no\_of\_students

#define nsub subject\_list::no\_of\_subjects

#define nsubstu subject::no\_of\_students

class subject{

static int no\_of\_students;

int code;

char name[11];

public:

void displayStudents(int);

friend class subject\_list;

friend class student;

friend class student\_list;

};

class subject\_list{

static int no\_of\_subjects;

subject list[100];

public:

void addSubject();

friend class student;

friend class student\_list;

};

class student{

int roll;

char name[31];

char phone[11];

subject\_list subs;

public:

friend class student\_list;

};

class student\_list{

static int no\_of\_students;

student list[100];

public:

void addStudent();

void displaySub(int);

void displayStudents(string);

};

int student\_list::no\_of\_students = 0;

int subject\_list::no\_of\_subjects=0;

int subject::no\_of\_students=0;

void student\_list::addStudent(){

if(nstu>100){

cout<<"STUDENT LIMIT REACHED. PLEASE CONTACT DEAN.\n";

return;

}

++nstu;

int ch;

list[nstu].roll=nstu;

cout<<"Enter Student Details:\n";

cout<<"Name: ";

cin>>list[nstu].name;

cout<<"Phone No.: ";

cin>>list[nstu].phone;

do{

list[nstu].subs.addSubject();

cout<<"Press 1 to add another subject, 0 to end: ";

cin>>ch;

}while(ch!=0);

cout<<"Alloted Roll of Student: A00"<<list[nstu].roll<<"\n";

}

void subject\_list::addSubject(){

if(nsub>100){

cout<<"SUBJECT LIMIT EXCEEDED. PLEASE CONTACT DEAN.\n";

return;

}

++nsub;

list[nsub].code=nsub;

cout<<"Enter Subject Name: ";

cin>>list[nsub].name;

cout<<"Subject Code: S00"<<list[nsub].code<<"\n";

}

void student\_list::displaySub(int a){

if(a<=0 || a>nstu){

cout<<"Invalid Roll Number!\n";

return;

}

cout<<"Subjects chosen by the student are:\n";

for(int i=1;i<=list[a].subs.no\_of\_subjects;i++)

cout<<list[a].subs.list[i].name<<"\n";

}

void student\_list::displayStudents(string a){

// if(a<=0 || a>nstu){

// cout<<"Invalid Subject Code!\n";

// return;

// }

cout<<"Students are:\n";

for(int i=1;i<=nstu;i++){

for(int j=1;j<=nsub;j++)

if(list[i].subs.list[j].name==a)

cout<<list[i].name<<"\n";

}

}

int main(){

student\_list ob;

int ch;

string stu\_code;

string sub;

do{

cout<<"\n\*\*\*STUDENT ADMISSION SYSTEM\*\*\*\n\n";

cout<<"1. ADD STUDENT.\n";

cout<<"2. DISPLAY STUDENT'S SUBJECTS.\n";

cout<<"3. DISPLAY STUDENTS AGAINST A SUBJECT.\n";

cout<<"4. EXIT.\n";

cout<<"Enter Your Choice: ";

cin>>ch;

switch(ch){

case 1:

ob.addStudent();

break;

case 2:

cout<<"Enter Roll (in format A00X): ";

cin>>stu\_code;

ob.displaySub(stu\_code[3] - '0');

break;

case 3:

cout<<"Enter Subject Name: ";

cin>>sub;

ob.displayStudents(sub);

break;

case 4:

exit(0);

break;

default:

cout<<"INVALID CHOICE!!\n";

}

}while(ch!=4);

return 0;

}

OUTPUT:

