**Assignment 3**

Q1.

CODE:

//to test macro in cpp

#include<iostream>

using namespace std;

#define max(a,b) (((a)>(b))?(a):(b)) //macro to find maximum from the arguments

int main(){

int a,b;

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

cout<<"MAXIMUM OF THE TWO IS "<<max(a,b)<<endl;

cout<<"FOR char\*:\n";

string c="HELLO";

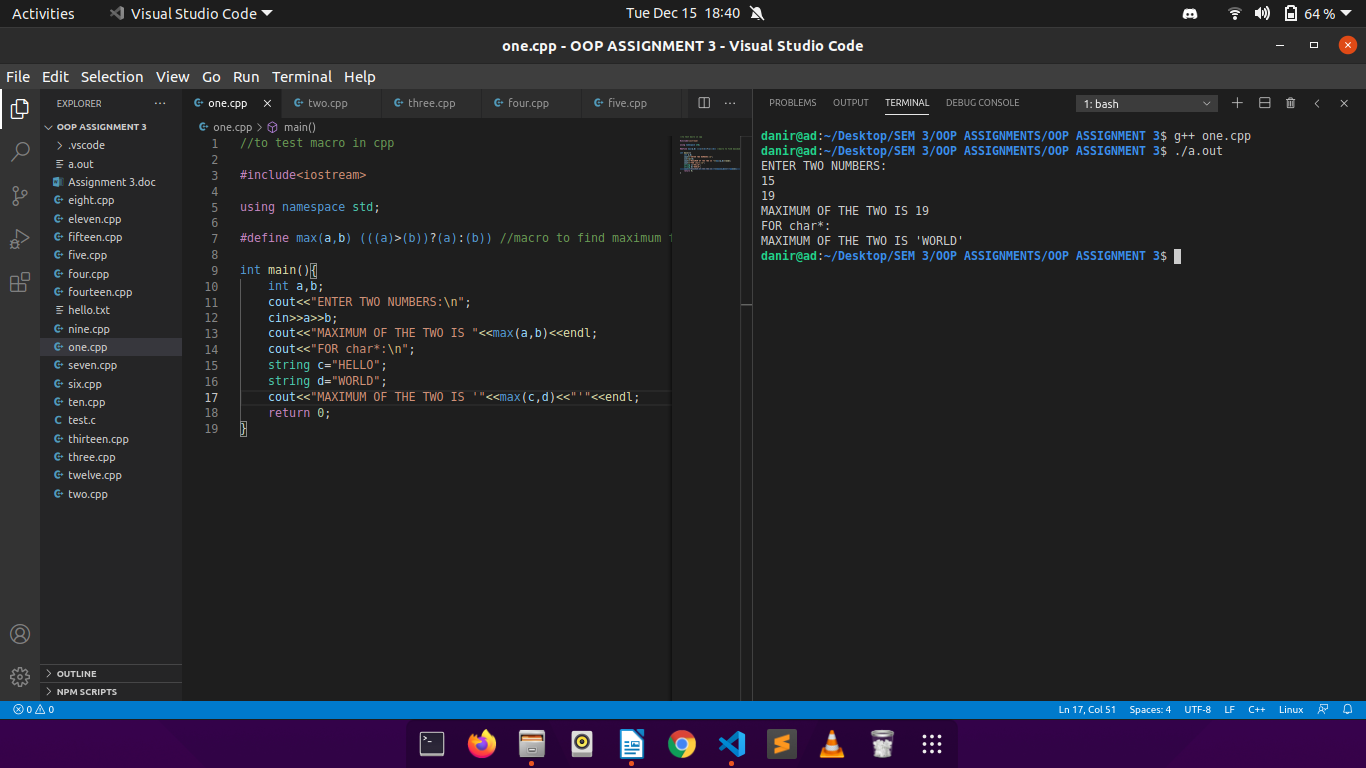
string d="WORLD";

cout<<"MAXIMUM OF THE TWO IS '"<<max(c,d)<<"'"<<endl;

return 0;

}

OUTPUT:



Q2.

CODE:

//inline functions in cpp

#include<bits/stdc++.h>

using namespace std;

/\*inline\*/ int product(int a, int b){

return a\*b;

}

int main(){

time\_t start, end;

time(&start);

ios\_base::sync\_with\_stdio(false);

int a,b;

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

cout<<"THEIR PRODUCT IS: "<< product(a,b) <<"\n";

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

cout<<"THEIR PRODUCT IS: "<< product(a,b) <<"\n";

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

cout<<"THEIR PRODUCT IS: "<< product(a,b) <<"\n\n";

time(&end);

double time\_taken = double(end - start);

cout << "Time taken by program is : " << fixed

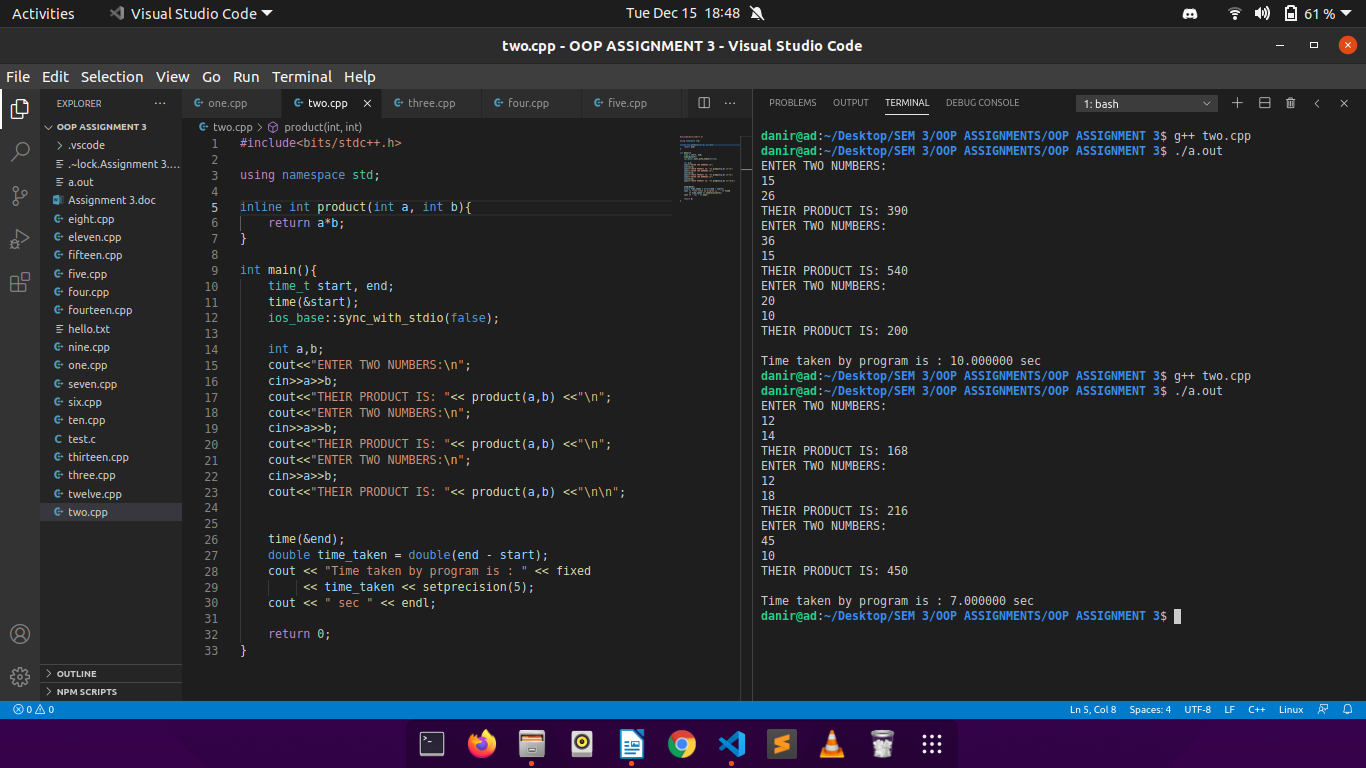
<< time\_taken << setprecision(5);

cout << " sec " << endl;

return 0;

}

OUTPUT:



The first output is without inline function and the second is with inline function.

Q3.

CODE:

//reference variables in cpp

#include<bits/stdc++.h>

using namespace std;

void swap(int &p, int &q){

int k;

k=p;

p=q;

q=k;

}

int main(){

int a,b;

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

cout<<"\nBEFORE SWAPPING...\n";

cout<<"FIRST NUMBER IS "<<a<<" and SECOND NUMBER IS "<<b<<"\n\n";

swap(a,b);

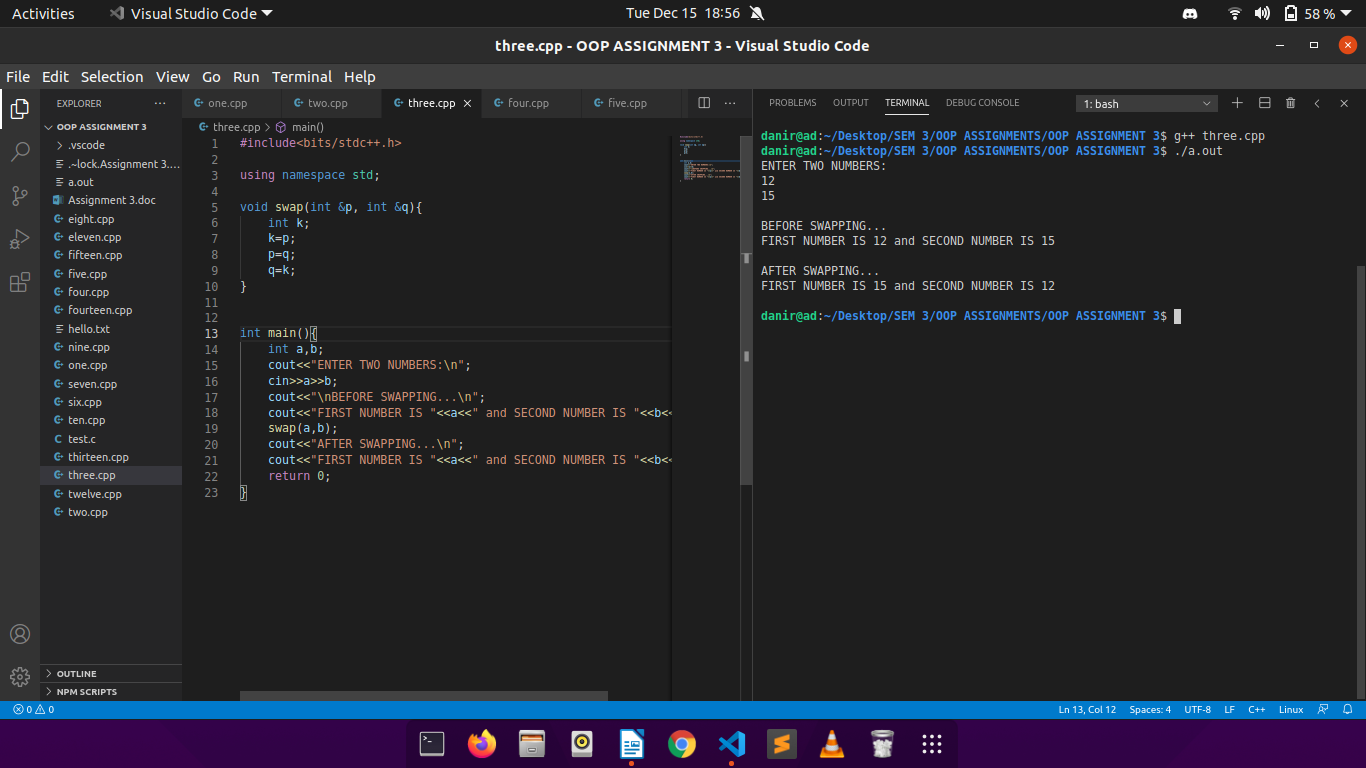
cout<<"AFTER SWAPPING...\n";

cout<<"FIRST NUMBER IS "<<a<<" and SECOND NUMBER IS "<<b<<"\n\n";

return 0;

}

OUTPUT:



Q4.

CODE:

//reference variables in cpp

#include<iostream>

using namespace std;

int& max(int &a, int &b){

if(a>b)

return a;

return b;

}

int main(){

int a,b;

cout<<"ENTER TWO NUMBERS:\n";

cin>>a>>b;

int x1=max(a,b);

int &x2=max(a,b);

cout<<"\nx1(of type int) and x2(of type int&):\n"<<x1<<"\t"<<x2<<endl;

cout<<"Values of a and b before modifying x1&x2:\n"<<a<<"\t"<<b<<endl<<endl;

x1++;x2++;

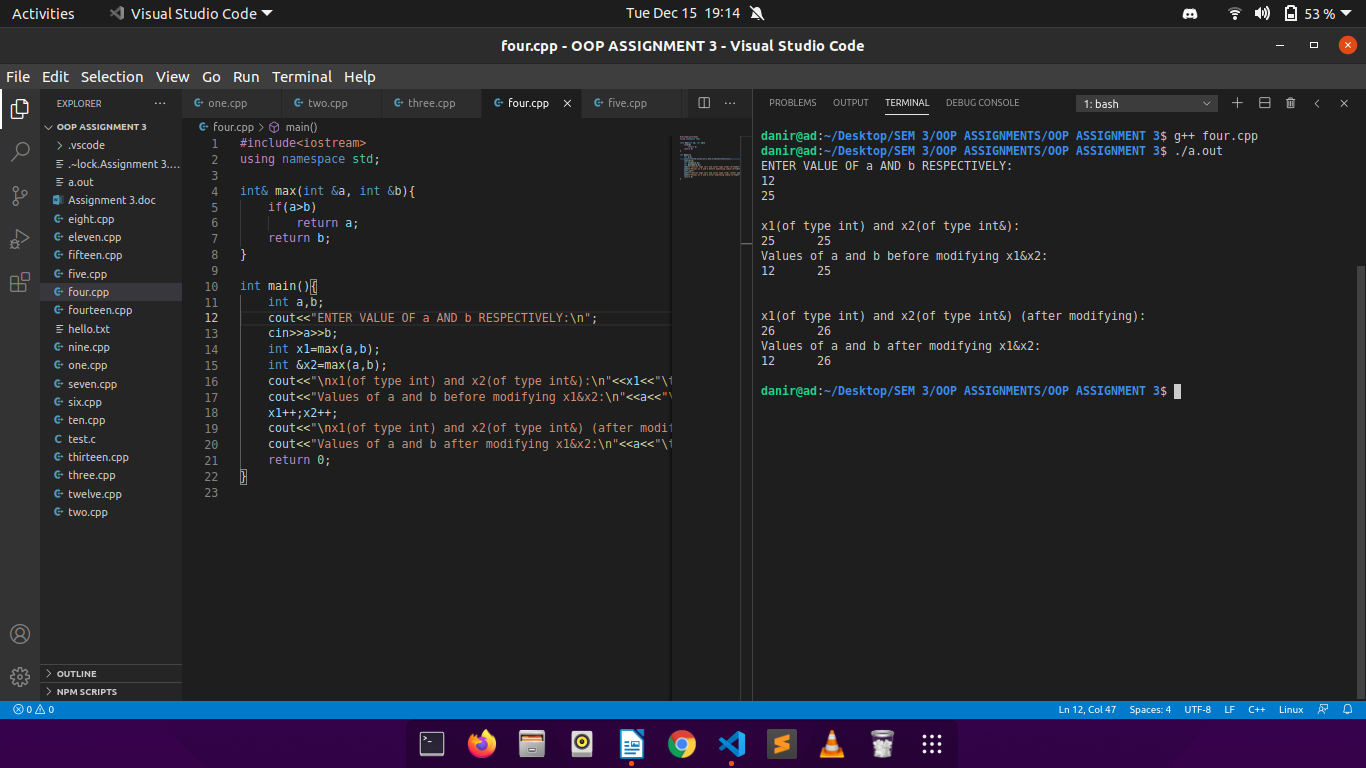
cout<<"\nx1(of type int) and x2(of type int&) (after modifying):\n"<<x1<<"\t"<<x2<<endl;

cout<<"Values of a and b after modifying x1&x2:\n"<<a<<"\t"<<b<<endl<<endl;

return 0;

}

OUTPUT:



Q5.

CODE:

//function with default values

#include<iostream>

using namespace std;

double tax(double income, int rate=10){

return ((income\*rate)/100);

}

int main(){

double income;

int rate,ch;

int k=1;

do{

cout<<"ENTER INCOME OF THE USER: ";

cin>>income;

cout<<"CALCULATE TAX WITH :\n";

cout<<"1. rate of 10%.\n";

cout<<"2. rate other than 10%.\n";

cout<<"ENTER YOUR CHOICE: ";

cin>>ch;

switch(ch){

case 1:

cout<<"\nThe tax amount is "<<tax(income)<<"."<<endl;

break;

case 2:

cout<<"\nENTER RATE OF TAX: ";

cin>>rate;

cout<<"\nThe tax amount is "<<tax(income)<<"."<<endl;

break;

default:

cout<<"WRONG CHOICE!";

}

cout<<"WANT TO CALCULATE TAX AGAIN? PRESS 1 TO CONTINUE, 0 TO ABORT: ";

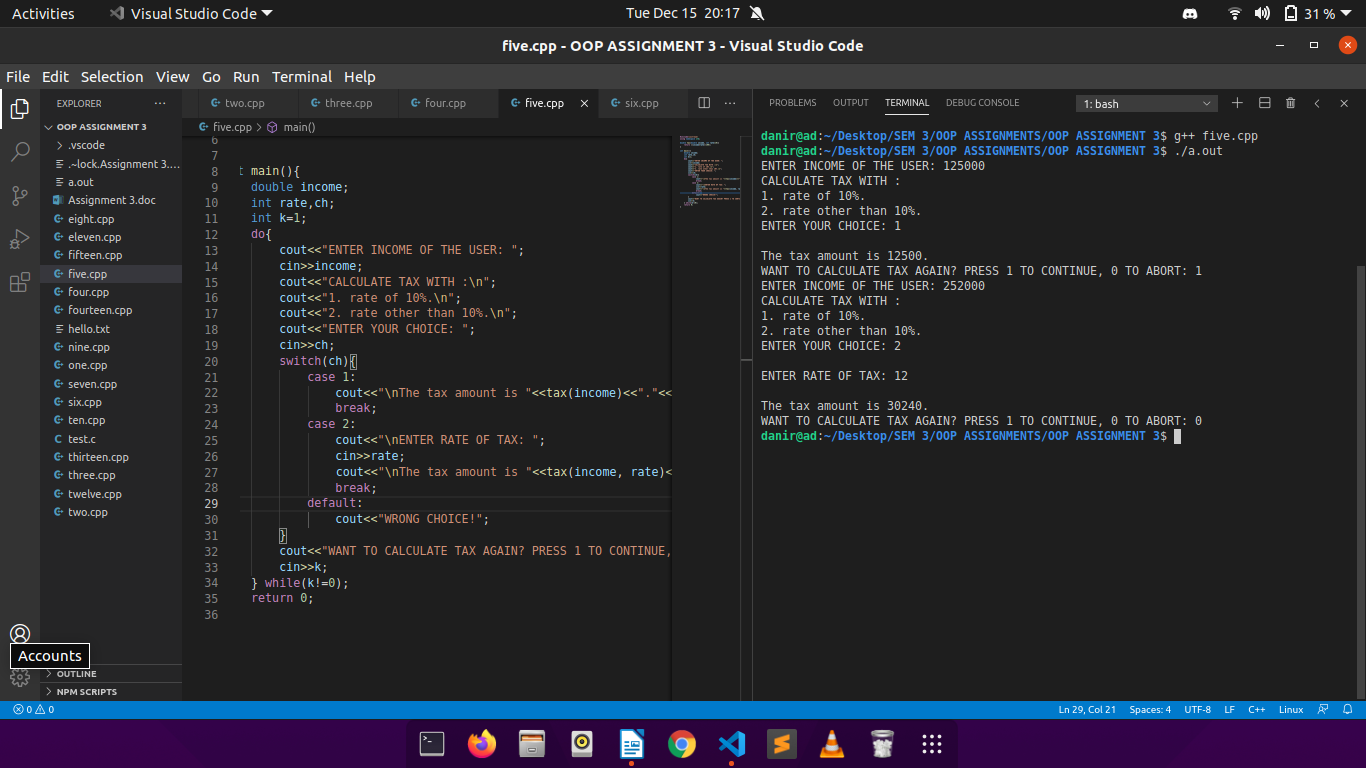
cin>>k;

} while(k!=0);

return 0;

}

OUTPUT:



Q6.

CODE:

//function overloading in cpp continued

#include<iostream>

using namespace std;

void f(int x){

cout<<"inside f(int)\n";

}

void f(float x){

cout<<"inside f(float)\n";

}

int main(){

int a;

char b;

float c;

double d;

cout<<"ENTER\n1. an integer\n2. a character\n3. a float\n4. a double\n\nin order as mentioned above:\n";

cin>>a>>b>>c>>d;

f(a);

f(b);

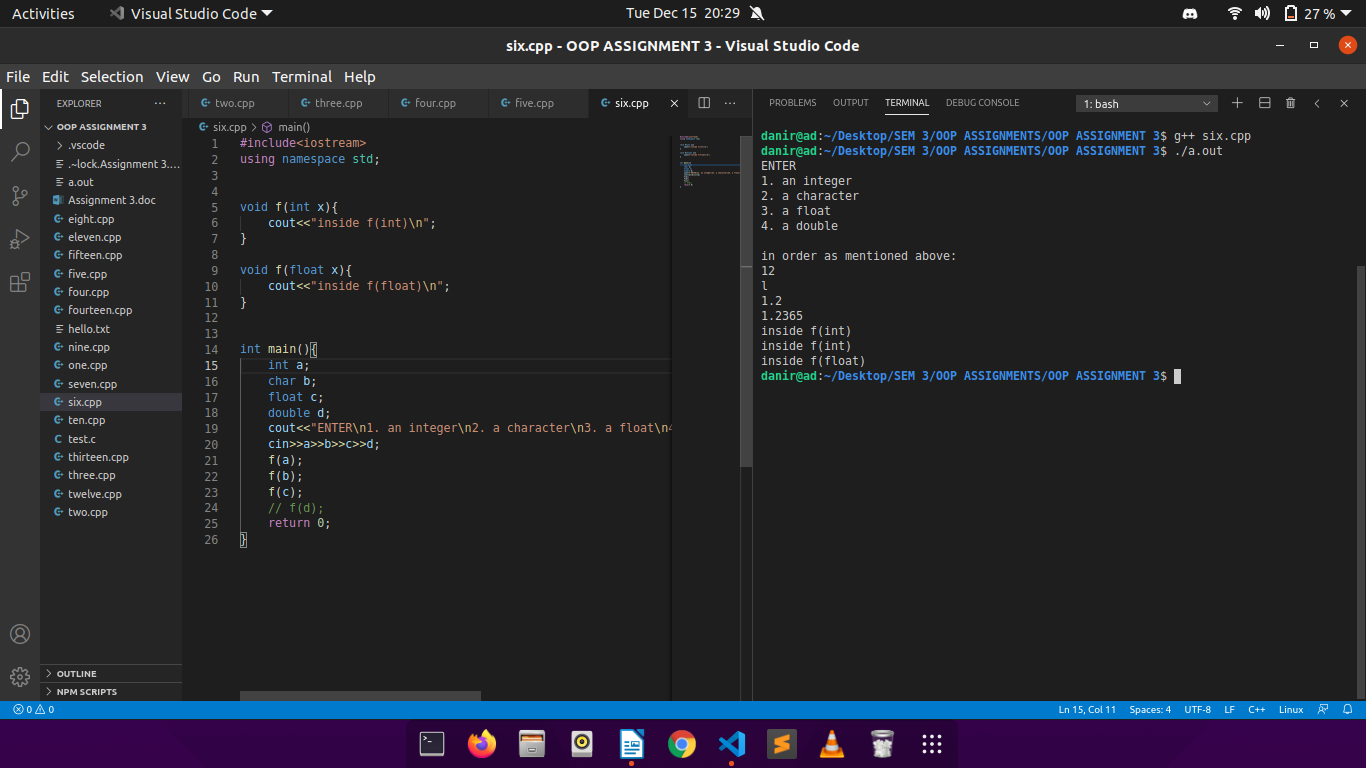
f(c);

// f(d); /*\*this is commented to avoid ambiguity error provided by the compiler\**/

return 0;

}

OUTPUT:



Q7.

CODE:

//function overloading in cpp

#include<iostream>

using namespace std;

void f(char c, int a){

cout<<"function with argument list (char, int).\n";

}

void f(int a, int b){

cout<<"function with argument list (int, int).\n";

}

int main(){

int a;

char c;

float x,y;

cout<<"ENTER AN INTEGER AND A CHARACTER RESPECTIVELY: ";

cin>>a>>c;

cout<<"ENTER TWO FLOAT VALUES: ";

cin>>x>>y;

f(a,c);

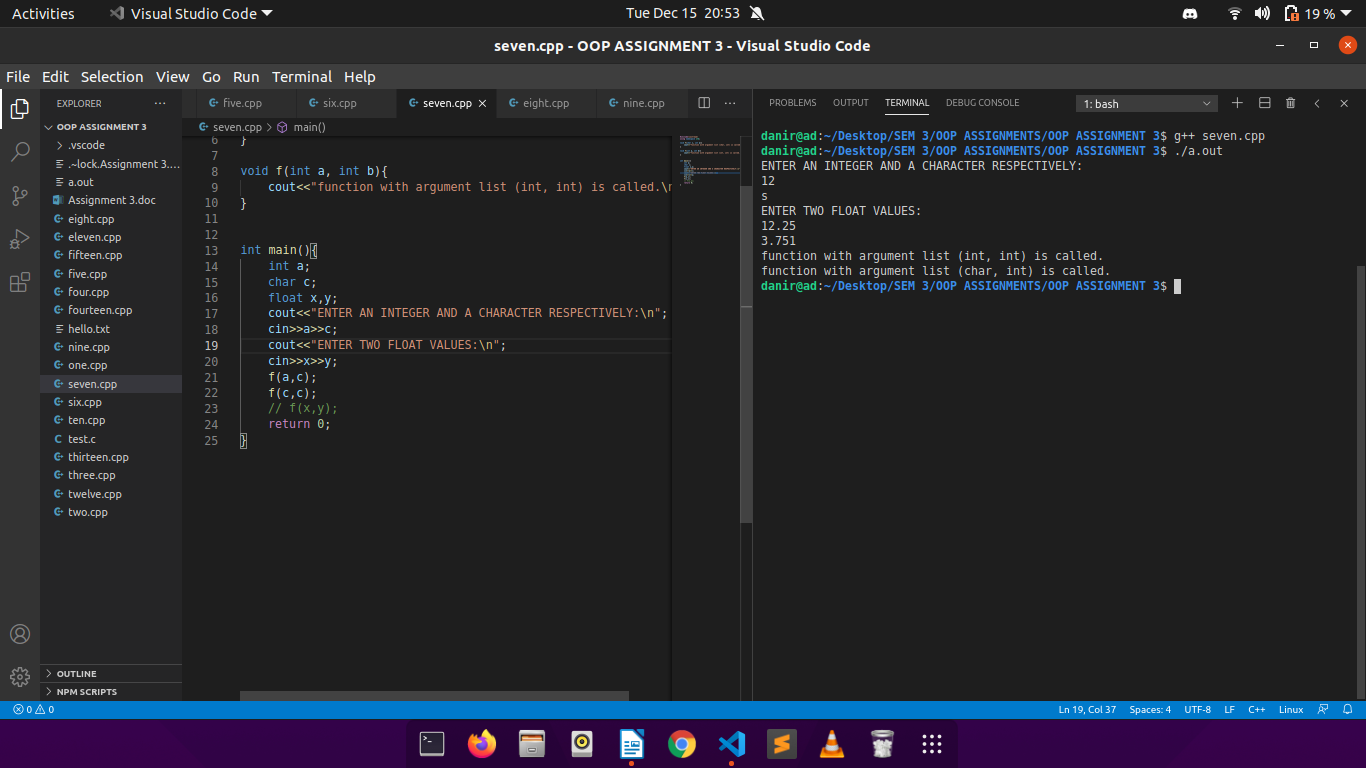
f(c,c);

// f(x,y); /*\*this is commented to avoid ambiguity error provided by the compiler\**/

return 0;

}

OUTPUT:



Q8.

CODE:

//structure in cpp

#include<iostream>

using namespace std;

struct student{

int roll;

char name[31];

int score;

void getdata();

void showdata();

};

void student::getdata(){

cout<<"\nENTER ROLL: ";

cin>>roll;

cout<<"ENTER NAME: ";

cin>>name;

cout<<"ENTER SCORE: ";

cin>>score;

}

void student::showdata(){

cout<<"\nROLL: "<<roll;

cout<<"\nNAME: "<<name;

cout<<"\nSCORE: "<<score<<"\n";

}

void modifyscore(int \*s){

\*s += 10;

}

int main(){

int n;

cout<<"ENTER NUMBER OF STUDENTS: ";

cin>>n;

student arr[n];

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

arr[i].getdata();

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

arr[i].showdata();

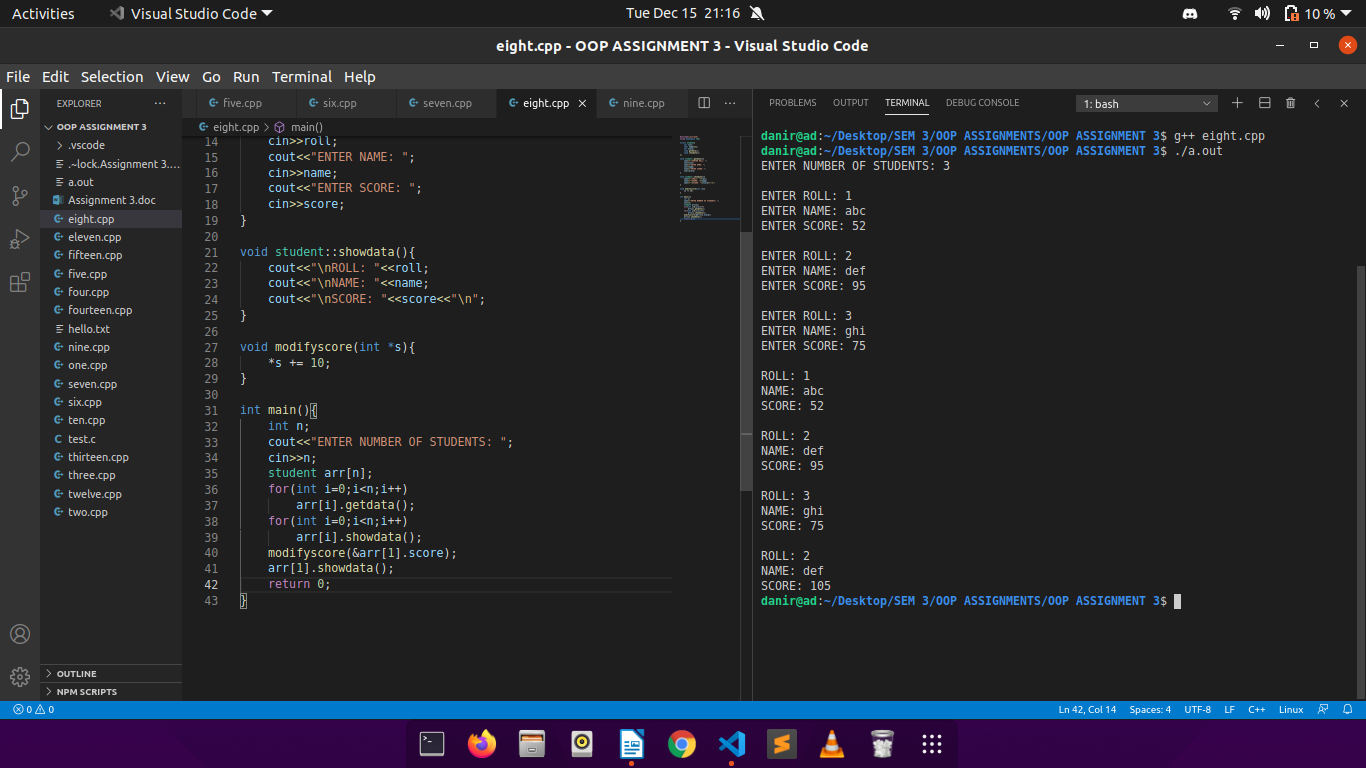
modifyscore(&arr[1].score);

arr[1].showdata();

return 0;

}

OUTPUT:



Q9.

CODE:

//changing time format according to user input

#include<iostream>

using namespace std;

class TIME {

int hour;

int min;

string time;

public:

void acceptTime(int);

void dispTime();

void addminute(int);

};

void TIME::acceptTime(int i){

if(i==0){

cout<<"ENTER TIME IN 12-hr FORMAT ( hh:mm:ssAM or hh:mm:ssPM where 01 <= hh <= 12 or 01 <= mm,ss <= 59 ):\n";

cin>>time;

}

else{

cout<<"ENTER TIME IN 24-hr FORMAT ( hh:mm:ss where 00 <= hh <= 23 or 01 <= mm,ss <= 59 ):\n";

cin>>time;

}

}

void TIME::dispTime(){

cout<<"THE TIME ENTERED BY THE USER IS: "<<time;

cout<<"\n\n";

}

void TIME::addminute(int j){

int k;

cout<<"\nENTER THE MINUTES TO BE ADDED: ";

cin>>k;

string t=this->time;

int min=((t[0] - '0') \* 10 + (t[1] - '0'))\*60 + ((t[3] - '0')\*10 + (t[4] - '0'));

min+=k;

cout<<"THE UPDATED TIME IS: ";

int hour = (min / 60) % 24;

int minu = min % 60;

int sec= (t[6]-'0')\*10 + (t[7]-'0');

if(j==1){

if (hour < 10)

cout<<0<<hour<<":";

else

cout<<hour<<":";

if (minu < 10)

cout << 0 << minu;

else

cout << minu;

cout<<":"<<sec<<endl;

}

else{

if(hour<12){

if (hour < 10)

cout<<0<<hour<<":";

else

cout<<hour<<":";

if (minu < 10)

cout << 0 << minu;

else

cout << minu;

cout<<":"<<sec<<"AM"<<endl;

}

else{

if(hour!=12)

hour=hour%12;

if (hour < 10)

cout<<0<<hour<<":";

else

cout<<hour<<":";

if (minu < 10)

cout << 0 << minu;

else

cout << minu;

cout<<":"<<sec;

if(t[8]=='A')

cout<<"PM\n";

else

cout<<"AM\n";

}

}

}

int main(){

TIME s;

int ch,k=0;

do{

cout<<endl<<"\*\*\*TIME STORING SYSTEM\*\*\*\n\n";

cout<<"1. ENTER TIME IN 24-hr FORMAT.\n";

cout<<"2. ENTER TIME IN 12-hr(AM/PM) FORMAT.\n";

cout<<"3. ADD MINNUTE TO EXISTING TIME.\n";

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

cout<<"5. EXIT.\n";

cout<<"\nENTER YOUR CHOICE: ";

cin>>ch;

switch(ch){

case 1:

s.acceptTime(1);

k=1;

break;

case 2:

s.acceptTime(0);

k=0;

break;

case 3:

s.addminute(k);

break;

case 4:

s.dispTime();

break;

// default:

// cout<<"WRONG CHOICE!!\n\n";

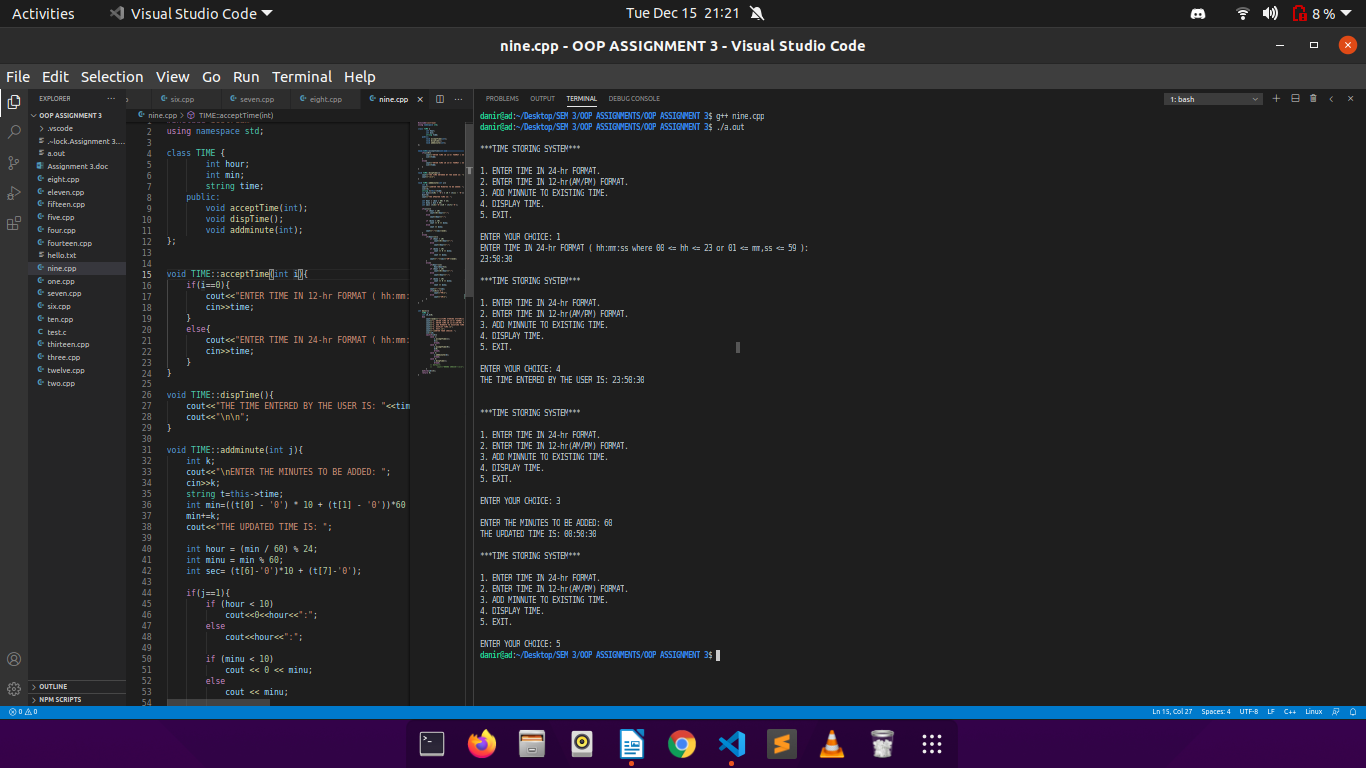
}

}while(ch<=4);

return 0;

}

OUTPUT:



Q10.

CODE:

//array implementation of stack

#include<iostream>

#define SIZE 100

using namespace std;

class STACK

{

int num[SIZE];

int top;

public:

STACK();

int push(int);

int pop();

int isEmpty();

int isFull();

void displayItems();

};

STACK::STACK(){

top=-1;

}

int STACK::isEmpty(){

if(top==-1)

return 1;

else

return 0;

}

int STACK::isFull(){

if(top==(SIZE-1))

return 1;

else

return 0;

}

int STACK::push(int n){

if(isFull()){

return 0;

}

++top;

num[top]=n;

return n;

}

int STACK::pop(){

int temp;

if(isEmpty())

return 0;

temp=num[top];

--top;

return temp;

}

void STACK::displayItems(){

int i;

cout<<"STACK is: ";

for(i=(top); i>=0; i--)

cout<<num[i]<<" ";

cout<<endl;

}

int main(){

STACK stk;

int ch, n,temp;

do{

cout<<endl;

cout<<"1. Push Item."<<endl;

cout<<"2. Pop Item."<<endl;

cout<<"3. Display Items."<<endl;

cout<<"4. EXIT."<<endl;

cout<<"Enter your choice: ";

cin>>ch;

switch(ch){

case 0: break;

case 1:

cout<<"Enter item to insert: ";

cin>>n;

temp=stk.push(n);

if(temp==0)

cout<<"STACK is FULL."<<endl;

else

cout<<temp<<" inserted."<<endl;

break;

case 2:

temp=stk.pop();

if(temp==0)

cout<<"STACK IS EMPTY."<<endl;

else

cout<<temp<<" is removed (popped)."<<endl;

break;

case 3:

stk.displayItems();

break;

case 4:

exit(0);

break;

default:

cout<<"An Invalid choice."<<endl;

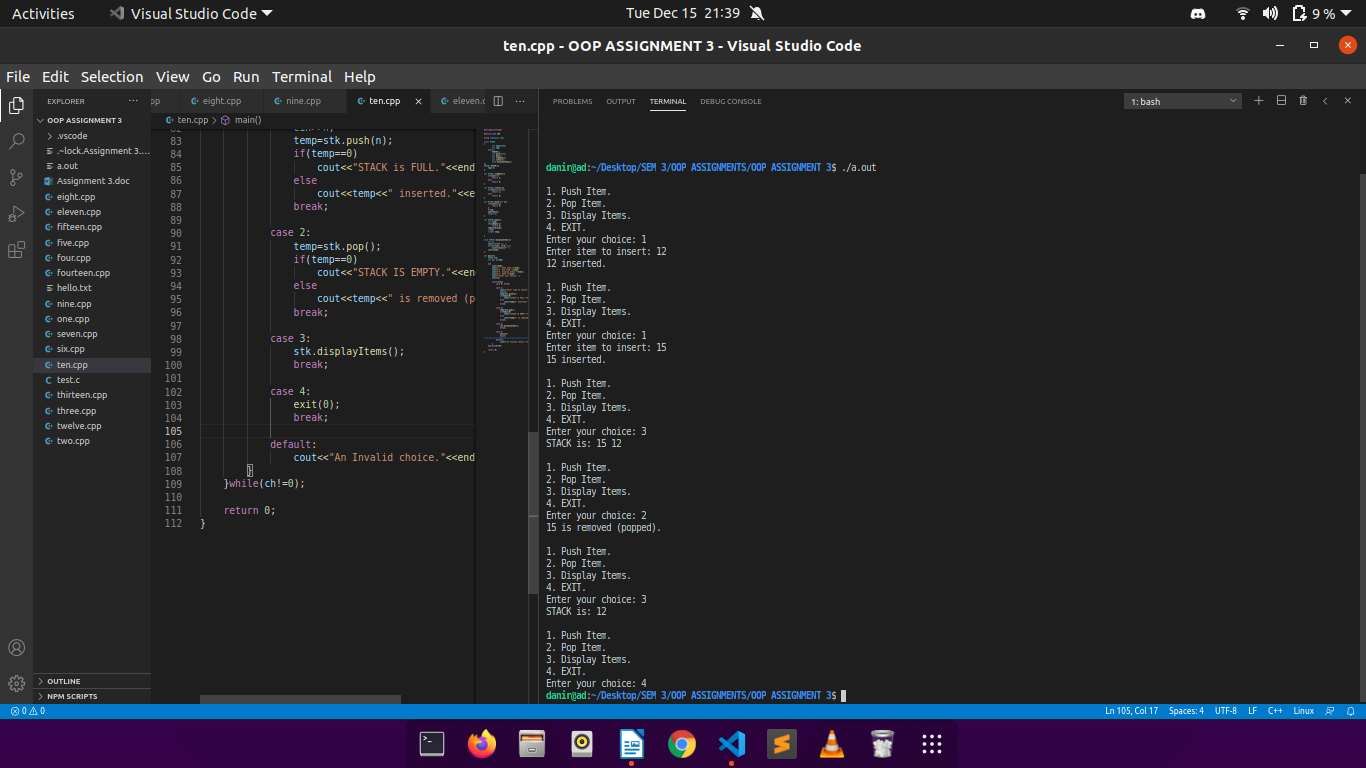
}

}while(ch!=0);

return 0;

}

OUTPUT:



Q11.

CODE:

//foundation classes in cpp

#include<iostream>

using namespace std;

class Applicant{

static int last\_roll;

int roll;

int score;

char name[31];

public:

void acceptData();

void showData() const;

static void showStudentCount(){

cout<<"The current number of students is "<<Applicant::last\_roll<<".\n";

}

};

int Applicant::last\_roll=0;

void Applicant::acceptData(){

Applicant::last\_roll++;

roll=Applicant::last\_roll;

cout<<"Enter Name of Student: ";

cin>>name;

cout<<"Enter Score of Student: ";

cin>>score;

}

void Applicant::showData() const{

cout<<"Roll: "<<roll<<endl;

cout<<"Name: "<<name<<endl;

cout<<"Score: "<<score<<endl;

}

int main(){

Applicant s;

int ch;

do{

cout<<"\n\*\*\*STUDENT DATABASE\*\*\*\n";

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

cout<<"2. SHOW LAST STUDENT DETAILS.\n";

cout<<"3. DISPLAY CURRENT COUNT OF STUDENTS.\n";

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

cout<<"\nENTER YOUR CHOICE: ";

cin>>ch;

switch(ch){

case 1:

s.acceptData();

break;

case 2:

s.showData();

break;

case 3:

Applicant::showStudentCount();

break;

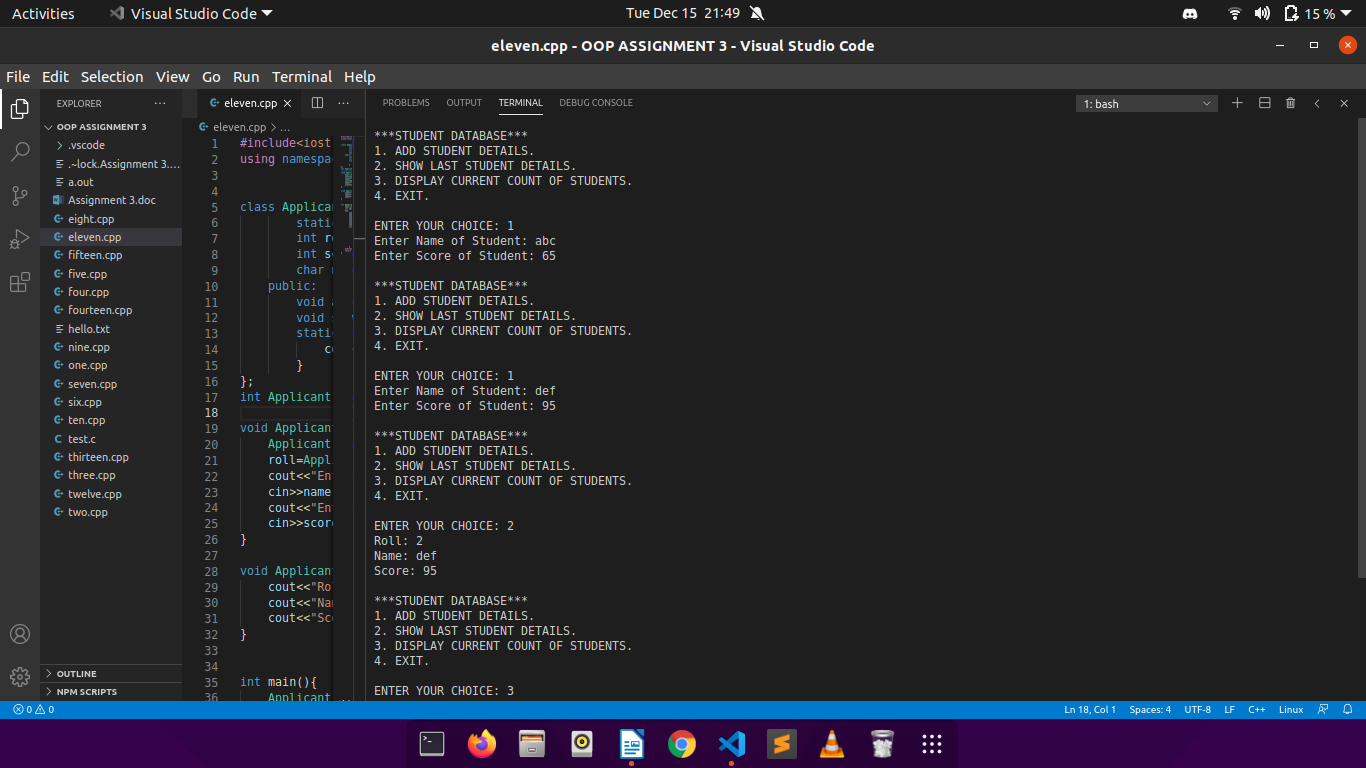
}

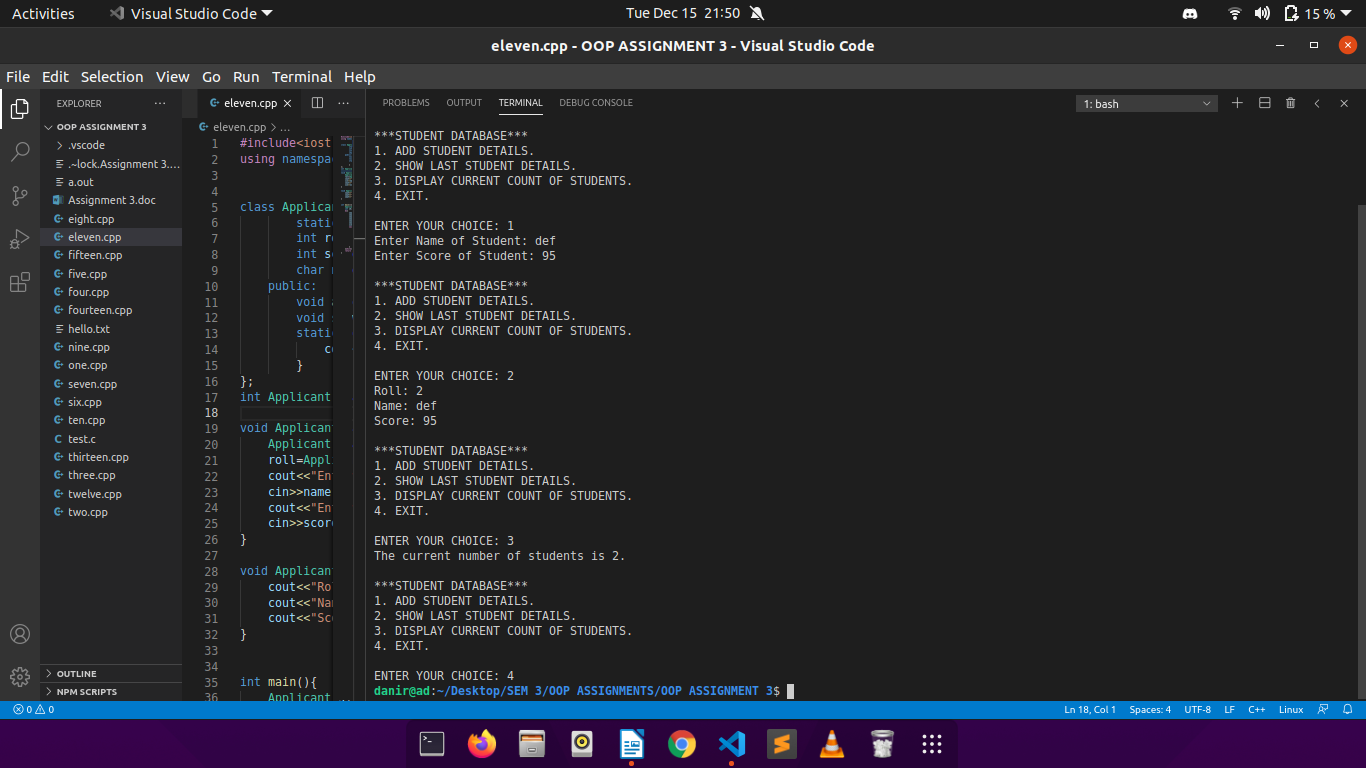
}while(ch!=4);

return 0;

}

OUTPUT:





Q12.

CODE:

//foundation classes in cpp

#include<iostream>

using namespace std;

class Student{

static int last\_roll;

int roll;

char name[31];

string course;

int dd;

int mm;

int yy;

int marks[5];

public:

Student()

{

roll = dd = mm = yy = -1;

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

marks[i] = -1;

}

}

void add();

void in\_marks();

bool check\_marks();

void marksheet();

static int numOfStudents(){

return last\_roll;

}

};

int Student::last\_roll = 0;

void Student::add(){

cout<<"\nEnter date of addmission:\n";

cout<<"Day: ";

cin>>dd;

cout<<"Month: ";

cin>>mm;

cout<<"Year: ";

cin>>yy;

cout<<"Enter Name of Student: ";

cin>>name;

cout<<"Enter Course Name: ";

cin>>course;

Student::last\_roll++;

roll=Student::last\_roll;

cout<<"Allotted Roll No. of "<<name<<" is: "<<roll;

}

void Student::in\_marks(){

cout<<"Enter Marks in 5 subjects:\n";

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

cout<<"Subject "<<i+1<<": ";

cin>>marks[i];

}

}

bool Student::check\_marks(){

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

if(marks[i]<0){

return 0;

}

}

return 1;

}

void Student::marksheet(){

if(!check\_marks()){

cout<<"MARKS NOT ENTERED!!\n";

return;

}

cout<<"\n\t\t\tMARKSHEET\n";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"NAME - "<<name<<"\n";

cout<<"Roll - "<<roll<<"\n";

cout<<"Dept - "<<course<<"\n";

cout<<"Date of admission(dd / mm / yyyy): "<<dd<<"/"<<mm<<"/"<<yy<<"\n";

cout<<"Marks: \n";

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

cout<<"Subject "<<(i + 1)<<": "<<marks[i]<<"\n";

}

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

class batch{

Student list[100];

public:

int enter\_roll();

void menu();

};

int batch::enter\_roll(){

cout<<"\nEnter roll of the student: ";

int r;

cin>>r;

if (r>0 && r<=Student::numOfStudents()) {

return r;

}

else{

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

return 0;

}

}

void batch::menu(){

int ch;

do {

cout<<"\n\n\t\*\*\*STUDENT MANAGEMENT SYSTEM\*\*\*\n";

cout<<"\n1. Admit Student.\n";

cout<<"2. Number of Admitted Students.\n";

cout<<"3. Enter Marks of an admitted student.\n";

cout<<"4. Print Marksheet of an admitted student.\n";

cout<<"5. Exit.\n";

cout<<"Enter Your Choice: ";

cin>>ch;

int r;

switch(ch){

case 1:

list[Student::numOfStudents()].add();

break;

case 2:

cout<<"\nNumber of students currently admitted: "<<Student::numOfStudents()<<"\n";

break;

case 3:

r=enter\_roll();

if(r){

list[r - 1].in\_marks();

}

break;

case 4:

r=enter\_roll();

if(r){

list[r - 1].marksheet();

}

break;

case 5:

exit(0);

break;

}

}while(ch!=5);

}

int main(){

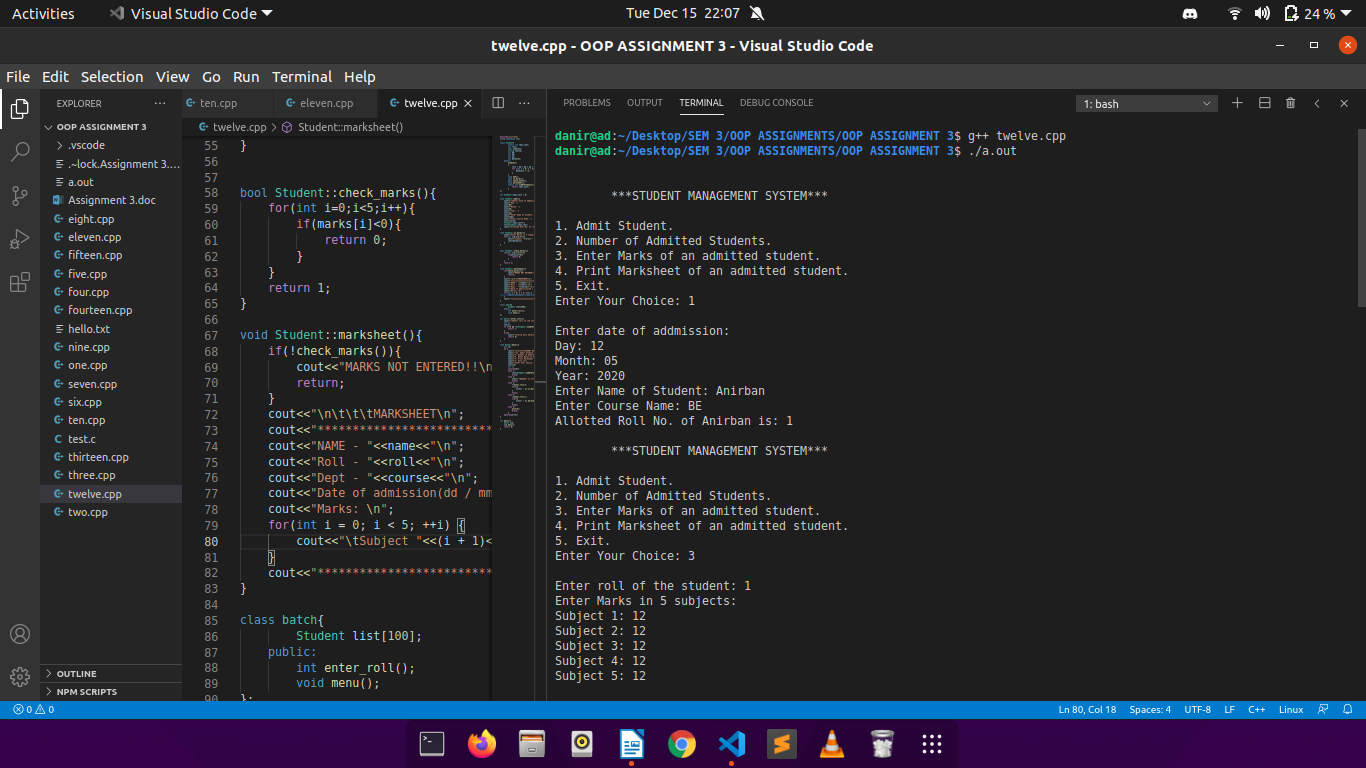
batch cse;

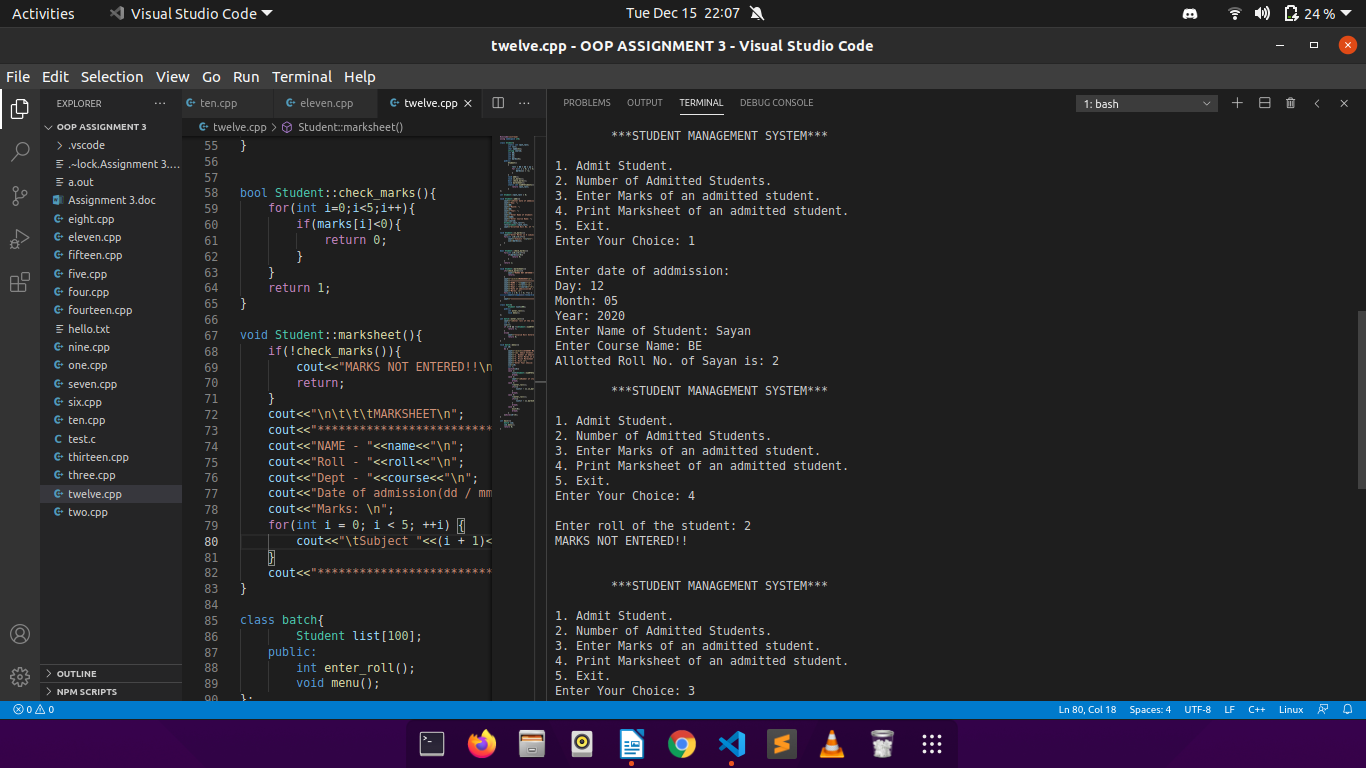
cse.menu();

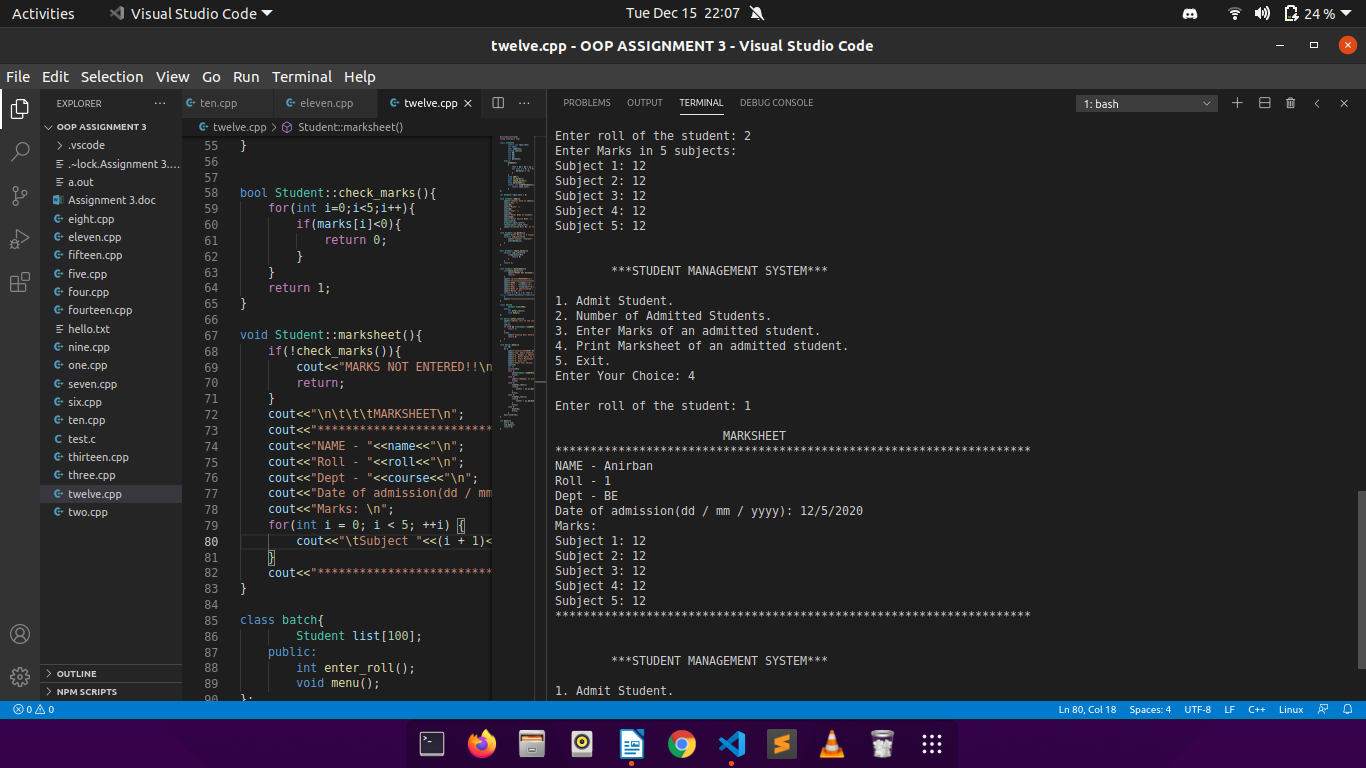
return 0;

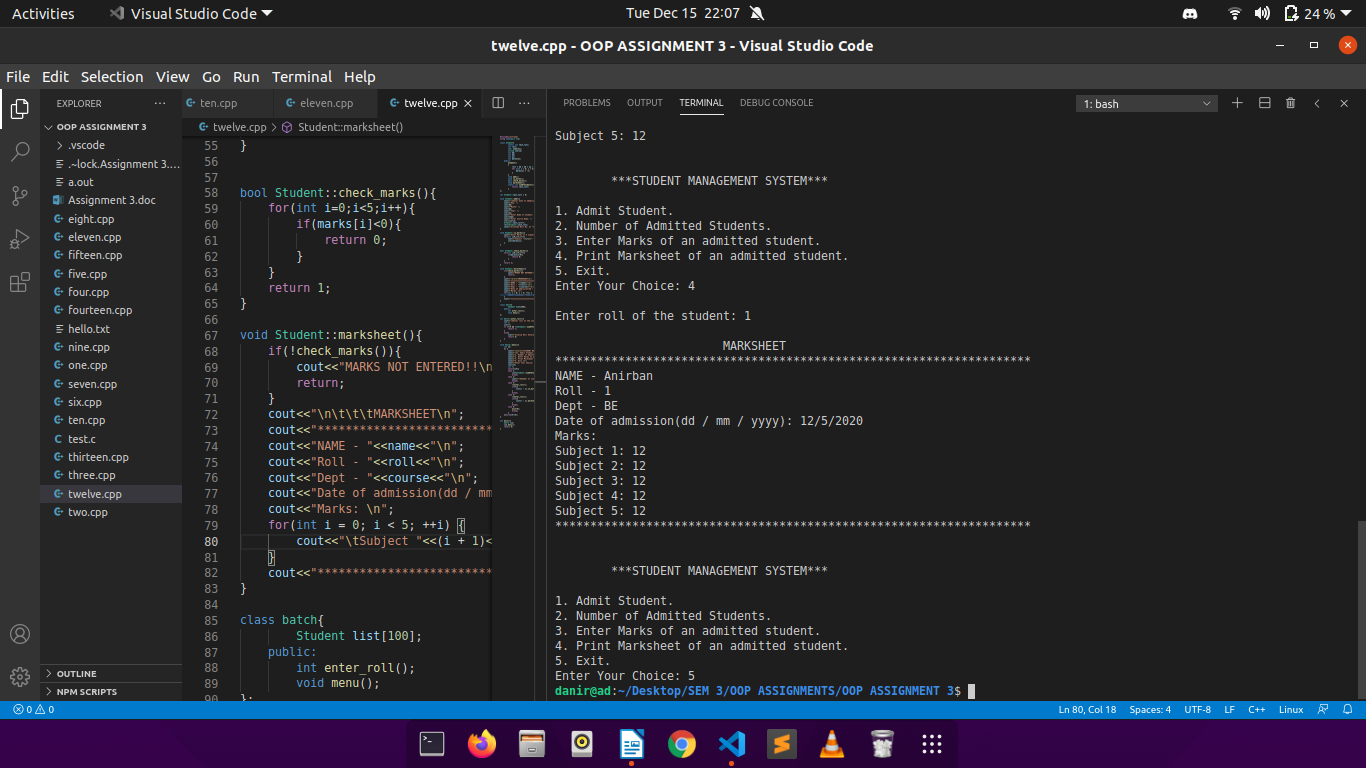
}

OUTPUT:









Q13.

CODE:

//linked list using cpp

#include<iostream>

using namespace std;

class Node{

public:

int info;

Node \*adr;

};

class LINKED\_LIST{

public:

Node \*head;

LINKED\_LIST(){

head=NULL;

}

void addNode(int);

void display();

void deleteNode(Node \*\*, int);

};

void LINKED\_LIST::addNode(int n){

Node\* newnode=new Node;

newnode->info=n;

newnode->adr=NULL;

if(head==NULL)

head=newnode;

else{

Node\* temp=head;

while(temp->adr!=NULL)

temp=temp->adr;

temp->adr=newnode;

}

}

void LINKED\_LIST::deleteNode(Node \*\*head, int key){

Node\* temp=\*head;

Node\* prev=NULL;

if(temp!=NULL && temp->info==key){

\*head=temp->adr;

delete temp;

return;

}

while (temp!=NULL && temp->info!=key) {

prev = temp;

temp = temp->adr;

}

if (temp == NULL)

return;

prev->adr = temp->adr;

delete temp;

}

void LINKED\_LIST::display(){

if(head==NULL)

cout<<"THE LIST IS EMPTY!"<<endl;

else{

Node\* temp=head;

cout<<"\nTHE CURRENT LIST LOOKS LIKE: ";

while(temp!=NULL){

cout<<temp->info<<" ";

temp=temp->adr;

}

cout<<endl;

}

}

int main(){

LINKED\_LIST\* list=new LINKED\_LIST;

int n,ch,key;

do{

cout<<"\n\*\*\*LINKED LIST\*\*\*\n\n";

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

cout<<"2. DELETE ELEMENT.\n";

cout<<"3. DISPLAY LIST.\n";

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

cout<<"Enter Your Choice: ";

cin>>ch;

switch(ch){

case 1:

cout<<"Enter the element to push: ";

cin>>n;

list->addNode(n);

break;

case 2:

cout<<"Enter the element to be popped: ";

cin>>key;

list->deleteNode(&list->head,key);

cout<<"\nDesired Node Deleted...\n";

list->display();

break;

case 3:

list->display();

break;

case 4:

cout<<"\*PROGRAM ENDED\*\n";

break;

default:

cout<<"Invalid Choice!!\n";

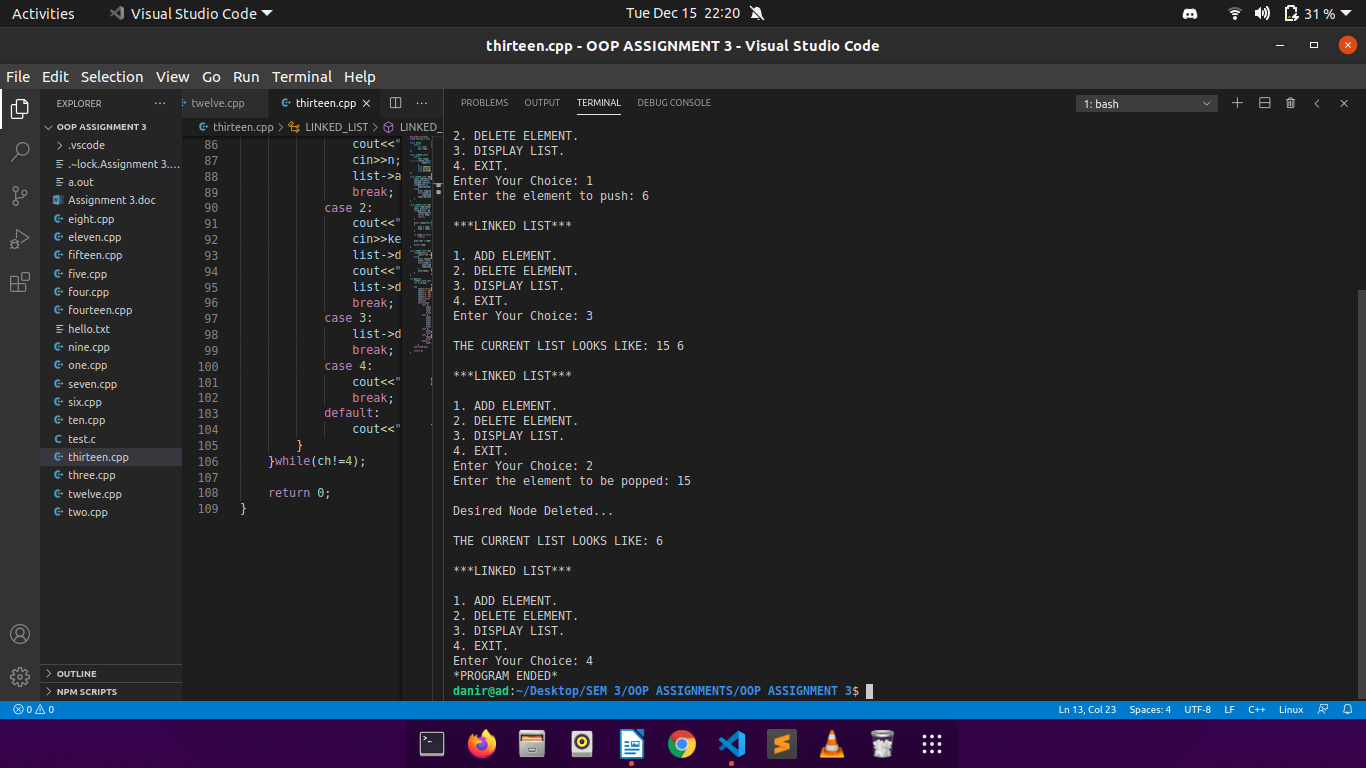
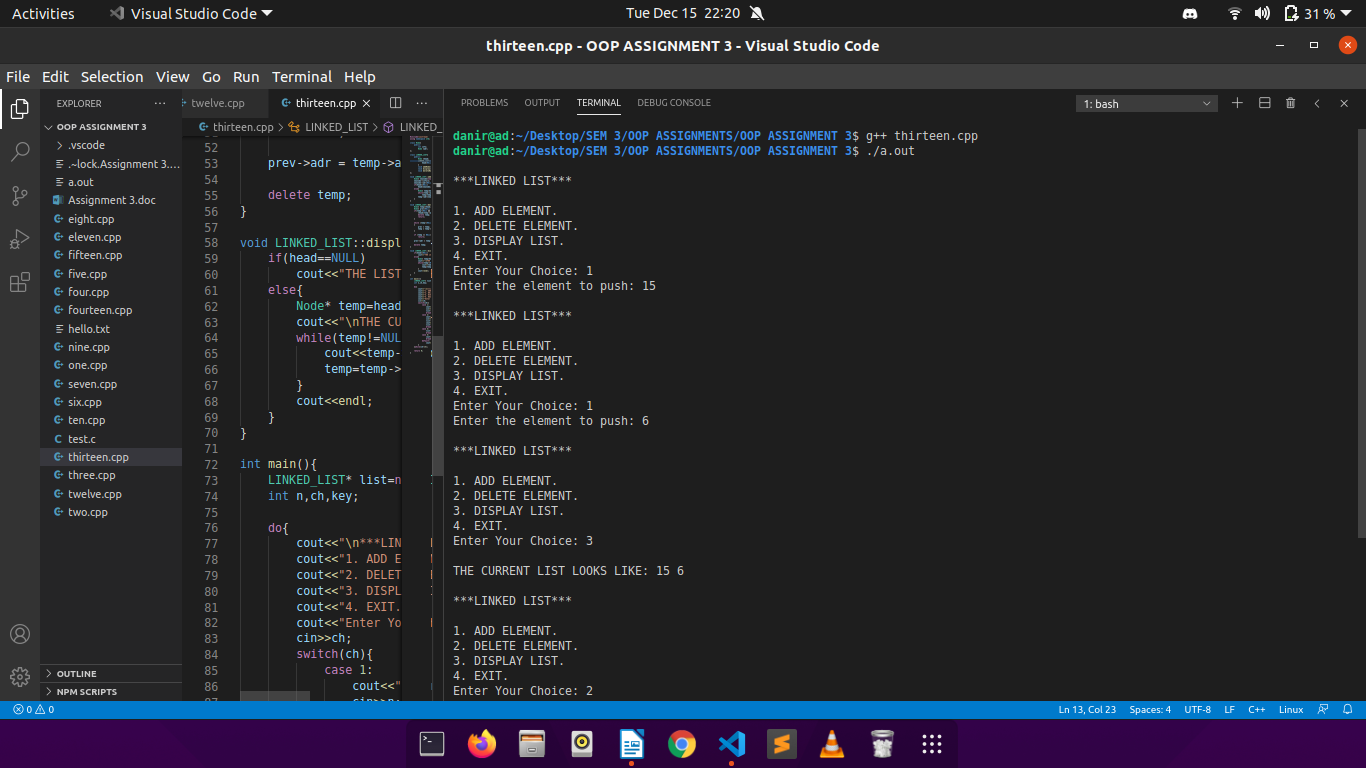
}

}while(ch!=4);

return 0;

}

OUTPUT:



Q14.

CODE:

//friend functions in cpp

#include<iostream>

using namespace std;

class sales;

class item{

static int last\_code;

int icode;

int qnty;

char iname[31];

int rate;

public:

void getdata();

void showdata();

void update\_rate();

void updatestock(sales);

int isavail();

static int numOfitems(){

return last\_code;

}

};

int item::last\_code=0;

void item::getdata(){

cout<<"Enter Item Name: ";

cin>>iname;

cout<<"Enter Quantity: ";

cin>>qnty;

cout<<"Enter Item Rate: ";

cin>>rate;

++item::last\_code;

icode=item::last\_code;

cout<<"Item Code: "<<icode<<"\n";

}

void item::showdata(){

cout<<"Item Code: "<<icode;

cout<<"\nItem Name: "<<iname;

cout<<"\nQuantity: "<<qnty;

cout<<"\nItem Rate: "<<rate<<"\n";

}

void item::update\_rate(){

cout<<"Enter new rate: ";

cin>>rate;

cout<<"Rate Changed Successfully!\n";

}

int item::isavail(){

if(qnty>0)

return 1;

return 0;

}

class sales{

item list[100];

int i;

int qntysold;

public:

void menu();

friend void item::updatestock(sales);

};

void item::updatestock(sales s){

qnty=qnty-s.qntysold;

}

void sales::menu(){

int ch;

do{

cout<<"\n\*\*\*ITEM SALES MANAGEMENT SYSTEM\*\*\*\n";

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

cout<<"2. ISSUE ITEM\n";

cout<<"3. MODIFY ITEM RATE\n";

cout<<"4. GET ITEM DETAILS\n";

cout<<"5. EXIT\n";

cout<<"\nEnter Your Choice: ";

cin>>ch;

int ic;

switch(ch){

case 1:

list[item::numOfitems()].getdata();

break;

case 2:

sales c;

cout<<"Enter code of Item to be issued: ";

cin>>c.i;

if(c.i<=item::numOfitems() && c.i>0){

cout<<"Enter quantity to be sold: ";

cin>>c.qntysold;

list[c.i-1].updatestock(c);

if(list[c.i-1].isavail())

cout<<"Item issued!\n";

else

cout<<"Out Of Stock!!\n";

}

else

cout<<"Invalid Item Code!\n";

break;

case 3:

cout<<"Enter code of Item to update rate: ";

cin>>ic;

list[ic-1].update\_rate();

break;

case 4:

cout<<"Enter code of Item to be issued: ";

cin>>ic;

list[ic-1].showdata();

break;

case 5:

exit(0);

break;

default:

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

}

}while(ch!=5);

}

int main(){

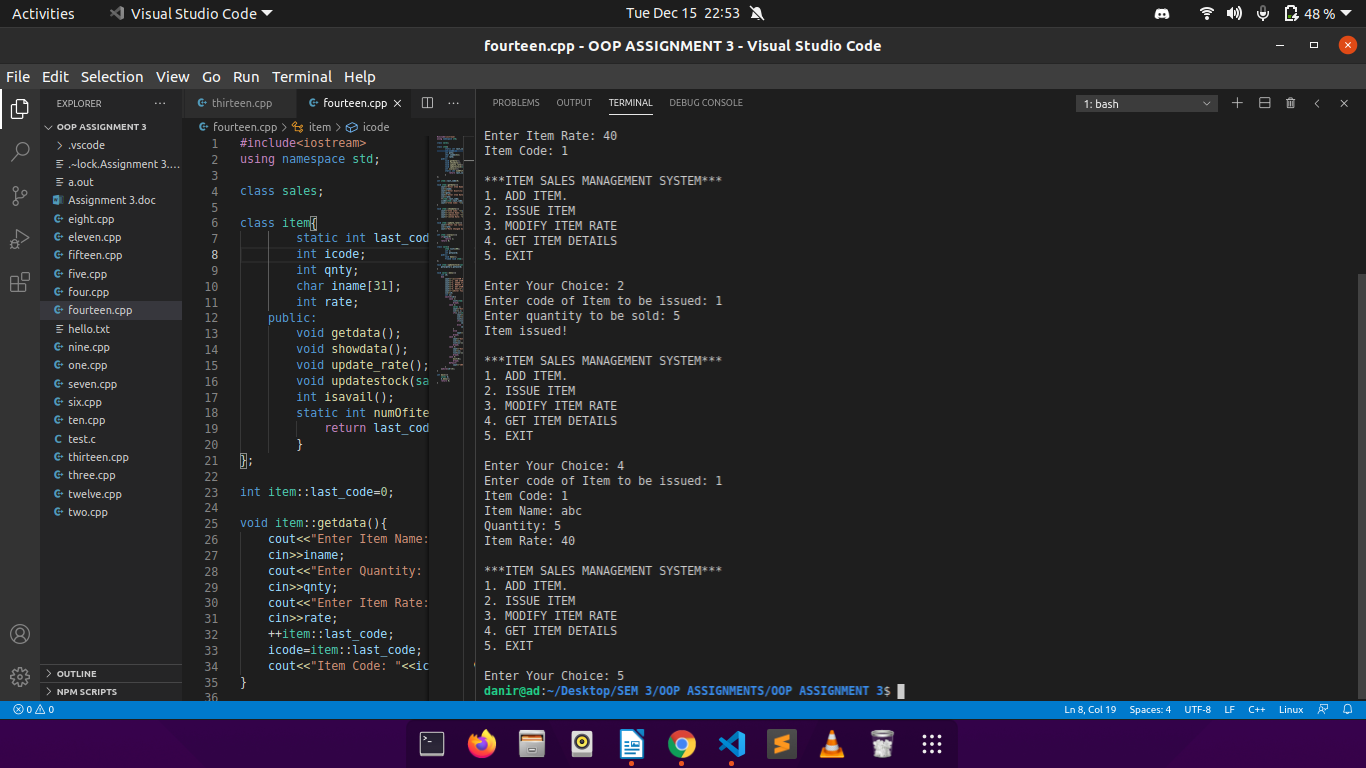
sales b;

b.menu();

return 0;

}

OUTPUT:



Q15.

CODE:

//friend functions in cpp

#include<iostream>

using namespace std;

class TRANSACTION;

class BALANCE{

static int last;

int accno;

int balance;

int dd;

int mm;

int yy;

public:

void input();

void show();

void update(TRANSACTION);

static int numOfitems(){

return last;

}

};

int BALANCE::last=0;

class TRANSACTION{

BALANCE b;

int acc;

int amt;

char type;

int d;

int m;

int y;

public:

void input();

friend void BALANCE::update(TRANSACTION);

void menu();

};

void BALANCE::input(){

cout<<"Enter Account Number: ";

cin>>accno;

cout<<"Enter Balance: ";

cin>>balance;

cout<<"Enter Date: ";

cout<<"\nDay: ";cin>>dd;

cout<<"Month: ";cin>>mm;

cout<<"Year: ";cin>>yy;

++BALANCE::last;

}

void BALANCE::show(){

cout<<"Acoount No.: "<<accno;

cout<<"\nBalance: "<<balance;

cout<<"\nLast Updated: "<<dd<<"/"<<mm<<"/"<<yy<<"\n";

}

void TRANSACTION::input(){

cout<<"Enter Account No.: ";

cin>>acc;

cout<<"Enter Transaction Type (W-WITHDRAWL, D-DEPOSIT): ";

cin>>type;

cout<<"Enter Amount: ";

cin>>amt;

cout<<"Enter Date: ";

cout<<"\nDay: ";cin>>d;

cout<<"Month: ";cin>>m;

cout<<"Year: ";cin>>y;

}

void BALANCE::update(TRANSACTION s){

if(s.type=='W')

balance=balance-s.amt;

else

balance=balance+s.amt;

dd=s.d;mm=s.m;yy=s.y;

}

void TRANSACTION::menu(){

int ch;

do{

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

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

cout<<"2. UPDATE ACCOUNT.\n";

cout<<"3. SHOW ACCOUNT DETAILS.\n";

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

cout<<"Enter YOur Choice: ";

cin>>ch;

switch(ch){

case 1:

b.input();

break;

case 2:

TRANSACTION c;

c.input();

b.update(c);

break;

case 3:

b.show();

break;

case 4:

exit(0);

break;

default:

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

}

}while(ch!=4);

}

int main(){

TRANSACTION k;

k.menu();

return 0;

}

OUTPUT:

