**QUESTION 1:**

**HEADER FILE:**

#pragma once

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class employee

{

private:

string id;

string name;

string department;

string grade;

const string bankaccount\_number;

public:

employee() ;

void input();

employee(string id, string name, string department, string grade, string bankaccount\_number);

string getacc() const;

void const display();

~employee();

};

**IMPLEMENTATION FILE:**

#include"Headser.h"

employee::employee() :bankaccount\_number("33102")

{

id = "\0";

name = "\0";

department = "\0";

grade = "\0";

}

void employee:: input()

{

cout << "enter name: ";

getline(cin, this->name);

cout << "enter id: ";

getline(cin, this->id);

cout << "enter grade: ";

getline(cin, this->grade);

if (grade == "0" || grade == "1" || grade == "2" || grade == "3" || grade == "4" || grade == "5" || grade == "6" || grade == "7" || grade == "8")

this->grade = grade;

else

this->grade = "-";

cout << "enter department: ";

getline(cin, this->department);

}

employee::employee(string id, string name, string department, string grade, string bankaccount\_number) : bankaccount\_number(bankaccount\_number)

{

this->id = id;

this->department = department;

if (grade == "0" || grade == "1" || grade == "2" || grade == "3" || grade == "4" || grade == "5" || grade == "6" || grade == "7" || grade == "8")

this->grade = grade;

else

this->grade = "-";

this->name = name;

}

string employee:: getacc() const

{

return bankaccount\_number;

}

void const employee:: display()

{

cout << setfill(' ') << setw(10) << name

<< setw(10) << grade

<< setw(15) << department

<< setw(10) << id

<< setw(15) << getacc() << endl;

}

employee::~employee()

{

cout << "destructor called ! " << endl;

}

**SOURCE FILE:**

#include"Headser.h"

int main()

{

employee arham("123", "Arham", "BSCS", "19", "33102");

employee\* e = new employee[3];

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

{

e[i].getacc();

e[i].input();

cout << endl;

}

cout << left << setfill(' ') << setw(10) << "name" << setw(10) << "grade" << setw(15) << "department" << setw(10) << "ID" << setw(15) << "bankaccount" << endl;

arham.display();

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

{

e[i].display();

}

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

{

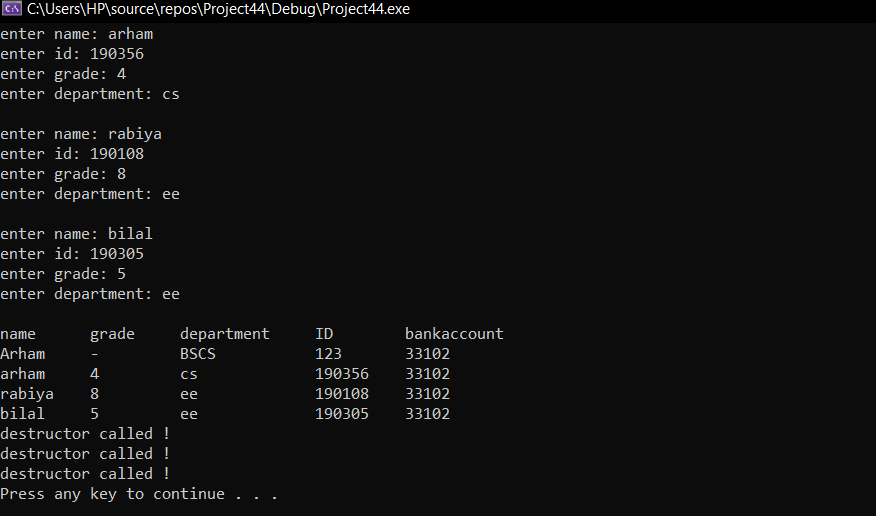
e[i].~employee();

}

system("pause");

return 0;

}



Question 2:

**HEADER FILE:**

#pragma once

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class person {

private:

string name;

const string dob;

const string cnic;

static int counter;

public:

person(string name, string str1, string str);

string getcnic();

string getdob();

void const display();

};

**IMPLEMENTATION FILE:**

#include"Headser.h"

person::person(string name, string str1, string str) :cnic(str1), dob(str)

{

this->name = name;

counter++;

}

string person::getcnic()

{

return cnic;

}

string person:: getdob()

{

return dob;

}

void const person::display()

{

cout << name << " " << dob << " " << cnic << endl;

cout << "the no of objects created = " << counter << endl;

}

**SOURCE FILE:**

#include"Headser.h"

int person::counter = 0;

int main() {

person p1("mr.x", "xxxxx-xxxxxxx-x", "1 january 2001");

p1.display();

person p2("sir hanan", "35201-xxxxxxx-x", "19 january 1995");

p2.display();

person p3("arham", "35202-xxxxxx-x", "3 rd january 2001");

p3.display();

system("pause");

return 0;

}

**OVER ALL A SINGLE CPP FILE CODE:**

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class person {

private:

string name;

const string dob;

const string cnic;

static int counter;

public:

person(string name,string str1,string str) :cnic(str1),dob(str)

{

this->name = name;

counter++;

}

string getcnic()

{

return cnic;

}

string getdob()

{

return dob;

}

void display() const

{

cout << name << " " << dob << " " << cnic << endl;

cout << "the no of objects created = "<<counter << endl;

}

};

int person::counter = 0;

int main() {

person p1("mr.x","xxxxx-xxxxxxx-x", "1 january 2001");

p1.display();

person p2("sir hanan", "35201-xxxxxxx-x", "19 january 1995");

p2.display();

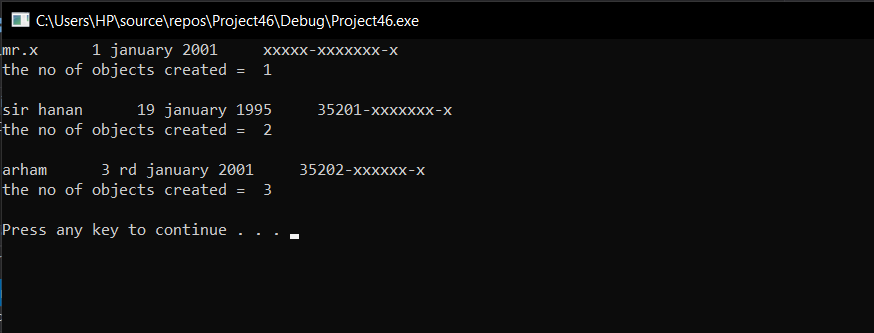
person p3("arham", "35202-xxxxxx-x", "3 rd january 2001");

p3.display();

system("pause");

return 0;

}



Question 4:

**Header file:**

#pragma once

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class software\_house

{

private:

double balance, interest\_rate, interest;

int trnascations;

static int counter;

public:

software\_house();

software\_house(double balnace, double interest\_rate);

void setinterest\_rate(double interest\_rate);

void makedeposit(double deposit);

void withdraw(double withdraw);

int calinterest();

int calinterest1();

int inccount();

int getcount();

double getinterest\_rate();

double getbalanve();

double getinterest();

void dispay();

};

**Implementation file:**

#include"Headser.h"

software\_house:: software\_house()

{

this->balance = 0;

this->interest\_rate = 4.5;

}

software\_house::software\_house(double balnace, double interest\_rate)

{

this->balance = balnace;

this->interest\_rate = interest\_rate;

}

void software\_house::setinterest\_rate(double interest\_rate)

{

this->interest\_rate;

}

void software\_house::makedeposit(double deposit)

{

this->balance = deposit;

}

void software\_house::withdraw(double withdraw)

{

if (withdraw > balance)

{

cout << "withdraw amount cannot be greater than current balance = " << endl;

}

else

balance = balance - withdraw;

}

int software\_house::calinterest()

{

interest = interest\_rate \* balance;

interest = interest / 100;

interest = balance + interest;

balance = interest;

return balance;

}

int software\_house::calinterest1()

{

interest = interest\_rate \* balance;

interest = interest / 100;

return interest;

}

int software\_house :: inccount() {

counter++;

return counter;

}

int software\_house::getcount() {

return counter;

}

double software\_house::getinterest\_rate()

{

return interest\_rate;

}

double software\_house::getbalanve()

{

return balance;

}

double software\_house::getinterest()

{

return interest;

}

void software\_house::dispay()

{

cout << "------------------------------MENU-----------------------------------" << endl;

cout << "1.DISPLAY CURRENT BALANCE = " << endl;

cout << "2.DISPLAY NUMBER OF TRANSACTIONS = " << endl;

cout << "3.DISPLAY INTEREST RATE FOR THIS PERIOD " << endl;

cout << "4.MAKE A WITHDRAWL " << endl;

cout << "5.ADD INTEREST RATE FOR THIS PERIOD = " << endl;

cout << "6. MAKE A DEPOSIT " << endl;

cout << "7.EXIT THE PROGRAM = " << endl;

int choice;

cout << "ENTER CHOICE = " << endl;

cin >> choice;

if (choice == 1)

{

cout << getbalanve();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

//else if (choice == 2)

// {

// cout << getcount();

// }

else if (choice == 3) {

cout << "INTEREST EARNED FOR THIS PERIOD= " << endl;

cout << calinterest1();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

else if (choice == 4) {

{

cout << "enter withdraw money= " << endl;

double x;

cin >> x;

withdraw(x);

counter++;

cout << getbalanve();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

}

else if (choice == 5)

{

{

cout << "INTEREST ADDED " << endl;

calinterest();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

}

else if (choice == 6)

{

cout << "enter deposit amount = " << endl;

double e;

cin >> e;

makedeposit(e);

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

else if (choice == 7)

{

exit;

}

}

**Source file:**

#include"Headser.h"

int software\_house::counter = 0;

int main() {

/\*software\_house o1;

o1.dispay();\*/

software\_house o2(0, 4.5);

int d = 1;

while (d != 0)

{

o2.dispay();

cout << "ENTER 0 FOR EXITING THR PROGRAM !" << endl;

cin >> d;

}

system("pause");

return 0;

}

**As a single cpp file :**

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

class software\_house

{

private:

double balance, interest\_rate, interest;

int trnascations;

static int counter;

public:

software\_house()

{

this->balance = 0;

this->interest\_rate = 4.5;

}

software\_house(double balnace, double interest\_rate)

{

this->balance=balnace;

this->interest\_rate = interest\_rate;

}

void setinterest\_rate(double interest\_rate)

{

this->interest\_rate;

}

void makedeposit(double deposit)

{

this->balance = deposit;

}

void withdraw(double withdraw)

{

if (withdraw > balance)

{

cout << "withdraw amount cannot be greater than current balance = " << endl;

}

else

balance = balance - withdraw;

}

int calinterest()

{

interest = interest\_rate \* balance;

interest = interest / 100;

interest = balance + interest;

balance=interest;

return balance;

}

int calinterest1()

{

interest = interest\_rate \* balance;

interest = interest / 100;

return interest;

}

int inccount() {

counter++;

return counter;

}

int getcount() {

return counter;

}

double getinterest\_rate()

{

return interest\_rate;

}

double getbalanve()

{

return balance;

}

double getinterest()

{

return interest;

}

void dispay()

{

cout << "------------------------------MENU-----------------------------------" << endl;

cout << "1.DISPLAY CURRENT BALANCE = " << endl;

cout << "2.DISPLAY NUMBER OF TRANSACTIONS = " << endl;

cout << "3.DISPLAY INTEREST RATE FOR THIS PERIOD " << endl;

cout << "4.MAKE A WITHDRAWL " << endl;

cout << "5.ADD INTEREST RATE FOR THIS PERIOD = " << endl;

cout << "6. MAKE A DEPOSIT " << endl;

cout << "7.EXIT THE PROGRAM = " << endl;

int choice;

cout << "ENTER CHOICE = " << endl;

cin >> choice;

if (choice == 1)

{

cout << getbalanve();

cout << endl;

cout <<"NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

//else if (choice == 2)

// {

// cout << getcount();

// }

else if (choice == 3) {

cout << "INTEREST EARNED FOR THIS PERIOD= " << endl;

cout<<calinterest1();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

else if (choice == 4) {

{

cout << "enter withdraw money= " << endl;

double x;

cin >> x;

withdraw(x);

counter++;

cout << getbalanve();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

}

else if (choice == 5)

{

{

cout << "INTEREST ADDED " << endl;

calinterest();

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

}

else if (choice == 6)

{

cout << "enter deposit amount = " << endl;

double e;

cin >> e;

makedeposit(e);

cout << endl;

cout << "NUMBER OF TIMES PROGRAM HAS TAKEN CHOICE =" << inccount() << endl;

}

else if (choice == 7)

{

exit;

}

}

};

int software\_house::counter = 0;

int main() {

/\*software\_house o1;

o1.dispay();\*/

software\_house o2(0,4.5);

int d=1;

while (d != 0)

{

o2.dispay();

cout << "ENTER 0 FOR EXITING THR PROGRAM !" << endl;

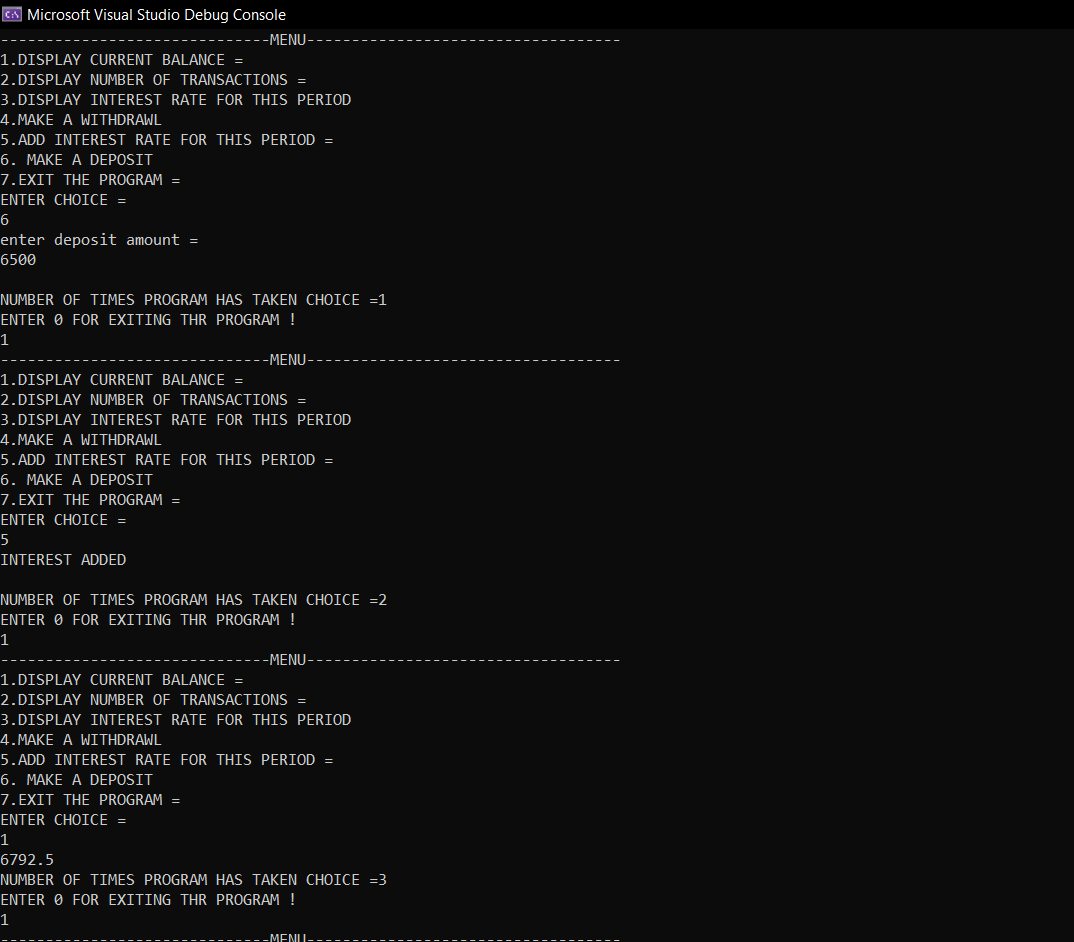
cin >> d;

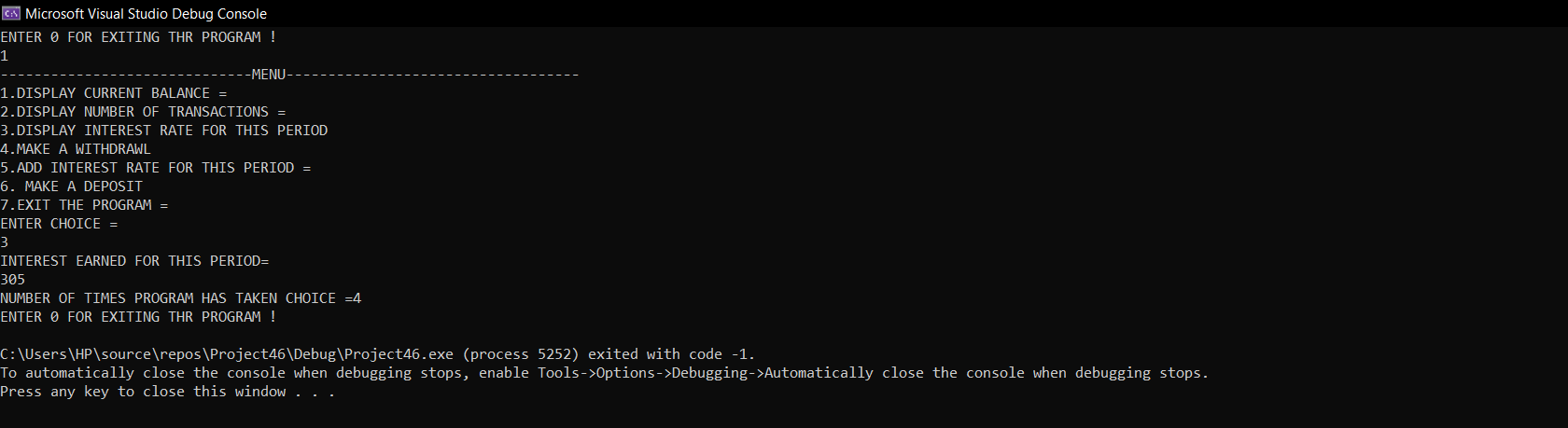
}

system("pause");

return 0;

}





Question 3:

#include<iostream>

#include<string>

#include<iomanip>

using namespace std;

int main() {

int x = 1;

while (x != -1)

{

cout << "1. Const Variables " << endl;

cout << "2. Constant Function Parameters" << endl;

cout << "3. Constant Return type " << endl;

cout << "4. Const Pointer " << endl;

cout << "5. Pointer to Const Variable" << endl;

cout << "6. Constant Data Members of Class " << endl;

cout << "7. Constant Member Function of Class " << endl;

int choice;

cin >> choice;

switch (choice)

{

case 1:

{

cout << " a variable whose value cannot be changed, is called a constand variable . we use keyword const for that " << endl;

cout << "For example = const double x = 65.66;" << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 2:

{

cout << "When you put const in front of a parameter, it means that it cannot be modified in the function" << endl;

cout << "for example = void g(const int\*) { function } " << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 3:

{

cout << "returns a constant value from a function " << endl;

cout << " for example = const int square(const int x) { return x \* x} " << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 4:

{

cout << "If we make a pointer const this means that the pointer will always point to the same address but we can change the value of that address. " << endl;

cout << "for example = int a=5; int \*const p=&a; " << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 5:

{

cout << "A pointer to a const value is a (non-const) pointer that points to a constant value" << endl;

cout << "for example = const int value = 5; " << endl;

cout << "const int\* ptr = &value; // this is okay, ptr is a non-const pointer that is pointing to a const int " << endl;

cout << "\*ptr = 6; " << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 6:

{

cout << "Data members are just variables declared inside a class. const data members are not assigned values during its declaration." << endl;

cout << "Const data members are assigned values in the constructor." << endl;

cout << " class A" << endl;

cout << "{" << endl;

cout << "const int x;" << endl;

cout << "public:" << endl;

cout << "A(int y)" << endl;

cout << "{" << endl;

cout << "x = y;" << endl;

cout << "}" << endl;

cout << "};" << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

case 7:

{

cout << "A const member function cannot change the value of any data member of the class and cannot call any member function which is not constant." << endl;

cout << "class A" << endl;

cout << "{" << endl;

cout << "public:" << endl;

cout << "int x;" << endl;

cout << "void func() const" << endl;

cout << " {" << endl;

cout << " x = 0;" << endl;

cout << " }" << endl;

cout << "};" << endl;

cout << "enter -1 to exit " << endl;

cin >> x;

break;

}

default:

break;

}

}

system("pause");

return 0;

}

