

Name: Adil Khan

Roll no: 2330-0030

Department: BS AI

Task no 1:

#include<iostream>

#include<string>

using namespace std;

class Staff {

protected:

int staffID;

public:

Staff(int id) {

staffID = id;

}

int getStaffID() const {

return staffID;

}

};

class Professor : public Staff {

protected:

int departmentID;

string departmentName;

public:

Professor(int id, int deptID, string name) : Staff(id) {

departmentID = deptID;

departmentName = name;

}

int getDepartmentID() const {

return departmentID;

}

string getDepartmentName() const {

return departmentName;

}

};

class VisitingProfessor : public Professor {

protected:

int no\_of\_course;

double salary\_per\_course;

public:

VisitingProfessor(int id, int deptID, string name, int course, double salary)

: Professor(id, deptID, name) {

no\_of\_course = course;

salary\_per\_course = salary;

}

double totalSalary() const {

return no\_of\_course \* salary\_per\_course;

}

void display() {

cout << "Staff ID: " << staffID << endl;

cout << "Department ID: " << departmentID << endl;

cout << "Department Name: " << departmentName << endl;

cout << "Number of courses: " << no\_of\_course << endl;

cout << "Per course salary: " << salary\_per\_course << endl;

cout << "Total salary: " << totalSalary() << endl;

}

};

int main() {

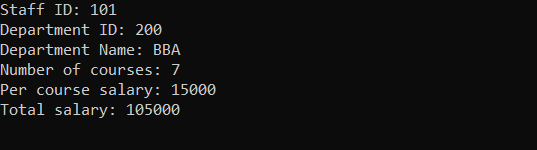
VisitingProfessor vp(101, 200, "BBA", 7, 15000);

vp.display();

return 0;

}

Output:



Task no 2:

#include <iostream>

using namespace std;

class Bank\_Account {

protected:

string accountHolder;

int accountNumber;

double balance;

public:

void createAccount(string name, int accNo, double initialBalance) {

accountHolder = name;

accountNumber = accNo;

balance = initialBalance;

}

void deposit(double amount) {

if (amount > 0) {

balance += amount;

cout << "Deposited: " << amount << "\nNew Balance: " << balance << endl;

} else {

cout << "Invalid deposit amount." << endl;

}

}

void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

cout << "Withdrawn: " << amount << "\nNew Balance: " << balance << endl;

} else {

cout << "Insufficient funds or invalid amount." << endl;

}

}

void display() {

cout << "Account Holder: " << accountHolder << "\nAccount Number: " << accountNumber

<< "\nBalance: " << balance << endl;

}

};

class SavingsAccount : public Bank\_Account {

public:

void withdraw(double amount) {

if ((balance - amount) >= 1000) {

balance -= amount;

cout << "Savings Withdrawn: " << amount << "\nNew Balance: " << balance << endl;

} else {

cout << "Cannot withdraw. Balance would fall below minimum (1000)." << endl;

}

}

};

class CurrentAccount : public Bank\_Account {

public:

void withdraw(double amount) {

if ((balance - amount) >= 1000) {

balance -= amount;

cout << "Current Withdrawn: " << amount << "\nNew Balance: " << balance << endl;

} else {

cout << "Cannot withdraw. Balance would fall below minimum (1000)." << endl;

}

}

};

int main() {

SavingsAccount sa;

CurrentAccount ca;

cout << "Creating Savings Account...\n";

sa.createAccount("Alice", 1001, 5000);

sa.display();

sa.withdraw(4500);

sa.withdraw(600);

cout << "\nCreating Current Account...\n";

ca.createAccount("Bob", 1002, 3000);

ca.display();

ca.withdraw(2200);

ca.withdraw(900);

return 0;

}

Output:

