## QUESTION 1:

**SOURCE CPP:**

#include"Header.h"

#include"Faculty.h"

#include"Student.h"

#include"Admin.h"

int main()

{

Faculty f;

f.namesetter("Ali");

f.agesetter(25);

f.gendersetter("Male");

f.fidsetter(123);

f.coursesetter("OOP");

f.display();

Student s;

s.namesetter("Ahmad");

s.agesetter(22);

s.gendersetter("Male");

s.sidsetter(456);

s.batchsetter(2021);

s.display();

Admin a;

a.namesetter("Abbas");

a.agesetter(30);

a.gendersetter("Male");

a.depsetter("CS");

a.display();

}

**PERSON CPP:**

#include"Header.h"

//Default Constructor

Person::Person() {}

//Setter Functions

void Person::namesetter(string name)

{

this->name = name;

}

void Person::agesetter(int age)

{

this->age = age;

}

void Person::gendersetter(string gender)

{

this->gender = gender;

}

//Getter Functions

string Person::namegetter()

{

return name;

}

int Person::agegetter()

{

return age;

}

string Person::gendergetter()

{

return gender;

}

**PERSON HEADER:**

#pragma once

#include<iostream>

using namespace std;

class Person

{

string name;

int age;

string gender;

public:

//Default Constructor

Person();

//Setter Functions

void namesetter(string name);

void agesetter(int age);

void gendersetter(string gender);

//Getter Functions

string namegetter();

int agegetter();

string gendergetter();

//Display Function

virtual void display() = 0;

}; **FACULTY HEADER:**

#pragma once

#include<iostream>

using namespace std;

class Faculty :public Person

{

int fid;

string course;

public:

Faculty();

//Setter Functions

void fidsetter(int fid);

void coursesetter(string course);

//Getter Functions

string coursegetter();

int fidgetter();

//Display Function

virtual void display();

}; **FACULTY CPP:**

#include"Faculty.h"

#include"Header.h"

Faculty::Faculty() {}

//Setter Functions

void Faculty::fidsetter(int fid)

{

this->fid = fid;

}

void Faculty::coursesetter(string course)

{

this->course = course;

}

//Getter Functions

string Faculty::coursegetter()

{

return course;

}

int Faculty::fidgetter()

{

return fid;

}

//Display Function

void Faculty::display()

{

cout << namegetter() << endl;

cout << agegetter() << endl;

cout << gendergetter() << endl;

cout << fidgetter() << endl;

cout << coursegetter() << endl << endl;

} **STUDENT HEADER:**

#pragma once

#include<iostream>

using namespace std;

class Student :public Person

{

int sid;

string course;

public:

Student();

//Setter Functions

void sidsetter(int sid);

void batchsetter(int batch);

//Getter Functions

string batchgetter();

int sidgetter();

//Display Function

virtual void display();

};

**STUDENT CPP:**

#include"Student.h"

#include"Header.h"

Student::Student() {}

//Setter Functions

void Student::sidsetter(int sid)

{

this->sid = sid;

}

void Student::batchsetter(int batch)

{

this->batchgetter = batch;

}

//Getter Functions

string Student::batchgetter()

{

return course;

}

int Student::sidgetter()

{

return sid;

}

//Display Function

void Student::display()

{

cout << namegetter() << endl;

cout << agegetter() << endl;

cout << gendergetter() << endl;

cout << sidgetter() << endl;

cout << batchgetter() << endl << endl;

} **ADMIN HEADER:**

#pragma once

#include<iostream>

using namespace std;

class Admin : public Student, public Faculty

{

string department;

public:

Admin();

//Setter Functions

void depsetter(string dep);

//Getter Functions

string depgetter();

//Display Function

virtual void display();

};

**ADMIN CPP:**

#include"Admin.h"

#include"Faculty.h"

#include"Student.h"

Admin::Admin() {}

//Setter Functions

void Admin::depsetter(string dep)

{

this->department = dep;

}

//Getter Functions

string Admin::depgetter()

{

return department;

}

//Display Function

void Admin::display()

{

cout << namegetter() << endl;

cout << agegetter() << endl;

cout << gendergetter() << endl;

cout << depgetter() << endl << endl;

}

# QUESTION 4:

#include<iostream>

#include<string>

#include<cstring>

using namespace std;

class Address

{

int house;

int street;

int apartment;

string city;

string state;

int postal;

public:

Address() //Default Constructor

{

this->house = 0;

this->street = 0;

this->apartment = 0;

this->city = " ";

this->state = " ";

this->postal = 0;

cout << "Default Constructor" << endl;

}

Address(int house, int street, int apartment, string city, string state, int postal) //Parametrized Constructor

{

this->house = house;

this->street = street;

this->apartment = apartment;

this->city = city;

this->state = state;

this->postal = postal;

cout << "Parametrized Constructor" << endl;

}

//Setter Funtions

void housesetter(int house)

{

this->house = house;

}

void streetsetter(int street)

{

this->street = street;

}

void apartsetter(int apar)

{

this->apartment = apar;

}

void citysetter(string ciyu)

{

this->city = city;

}

void statesetter(string state)

{

this->state = state;

}

void postalsetter(int postal)

{

this->postal = postal;

}

//Getter Functions

int housegetter()

{

return house;

}

int streetgetter()

{

return street;

}

int apartgetter()

{

return apartment;

}

string citygetter()

{

return city;

}

string stategetter()

{

return state;

}

int postalgetter()

{

return postal;

}

//Input Function

void input()

{

string str;

int i;

cout << "Enter house: " << endl;

cin >> i;

housesetter(i);

cout << "Enter STREET: " << endl;

cin >> i;

streetsetter(i);

cout << "Enter apartment: " << endl;

cin >> i;

apartsetter(i);

cout << "Enter city: " << endl;

cin >> str;

citysetter(str);

cout << "Enter state: " << endl;

cin >> str;

statesetter(str);

cout << "Enter postal: " << endl;

cin >> i;

postalsetter(i);

}

//Output Function

void display()

{

cout << "Street : " << streetgetter() << endl;

cout << "City: " << citygetter() << " State: " << stategetter() << " Postal: " << postalgetter() << endl;

}

//Operator Overloading

void operator !()

{

if (house != 0, street != 0, city != " ", postal != 0)

{

cout << "Address is Valid." << endl;

}

else

{

cout << "Address is not Valid." << endl;

}

}

void operator - (Address add)

{

if (add.postal<100)

{

cout << "Address comes before" << endl;

}

else

{

cout << "Address comes after." << endl;

}

}

~Address()

{

cout << "I am Destructor" << endl;

}

};

class PersonBio : public Address

{

public:

string name;

string proffession;

int age;

Address add;

PersonBio() //Default Constructor

{

this->name = " ";

this->proffession = " ";

this->age = 0;

cout << "Default Constructor !" << endl;

}

PersonBio(string name, string proffession, int age) //Parametrized Constructor

{

this->name = name;

this->proffession = proffession;

this->age = age;

cout << "Parametrizedd Constructor !" << endl;

}

//Setter Funtions

void namesetter(string name)

{

this->name = name;

}

void agesetter(int age)

{

this->age = age;

}

void Profsetter(string prof)

{

this->proffession = prof;

}

//Getter Functions

string namegetter()

{

return name;

}

int agegetter()

{

return age;

}

string profgetter()

{

return proffession;

}

};

int main()

{

Address a1, a2, a3;

a1.housesetter(1);

a1.streetsetter(2);

a1.apartsetter(3);

a1.citysetter("Sgd");

a1.statesetter("punjab");

a1.postalsetter(40100);

a1.display();

a2.housesetter(4);

a2.streetsetter(5);

a2.apartsetter(6);

a2.citysetter("Lhr");

a2.statesetter("punjab");

a2.postalsetter(41000);

a2.display();

a3.housesetter(7);

a3.streetsetter(8);

a3.apartsetter(9);

a3.citysetter("Fsl");

a3.statesetter("punjab");

a3.postalsetter(40001);

a3.display();

a1 = a2;

a1.display();

a3.input();

a3.display();

a1 - a3;

system("pause") ;

}

# OUTPUT:

