## Base Class (Person):-

Data members: name (string), age (int)

Member functions: getDetails(), a virtual function to print basic person details

Derived Class (Student): (Single Inheritance)

```
#include <iostream>

#include <string>

class Person {

protected:

    std::string name;

    int age;

public:                         //constructor

    Person(const std::string& n, int a) : name(n), age(a) {}

    void printDetails() {                                                    //methods to
print details

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

        std::cout << "Age: " << age << std::endl;

    }

};

class Details : public Person {                                    // Derived class Details
inherits from Person

public:

    Details(const std::string& n, int a) : Person(n, a) {}

};


int main() {                        // main function

    Details person1("Arjun", 20);           // Create an object of the derived class
```

```
    person1.printDetails();          // Call the printDetails method
    return 0;
}
```

## Inherits from Person :-

Data members: studentId (int), major (string)

Member functions:

setMajor(string) to set the student's major

getMajor() to retrieve the major

Override getDetails() to include student-specific information

Derived Class (Faculty): (Single Inheritance)

```
#include <iostream>
#include <string>
using namespace std;
class Person {
protected:
    string name;
    int age;
public:
    Person(string n, int a) : name(n), age(a) {}
    void setName(string n) { name = n; }
    void setAge(int a) { age = a; }
    string getName() { return name; }
    int getAge() { return age; }
    virtual void getDetails() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
```

```
};
class Student : public Person {                    // Derived class Student
    private:
    int studentId;
    string major;
public:
    Student(string n, int a, int sid) : Person(n, a), studentId(sid) {}
    void setMajor(string m) { major = m; }
    string getMajor() { return major; }
    void getDetails() override {
        Person::getDetails();
        cout << "Student ID: " << studentId << endl;
        cout << "Major: " << major << endl;
    }
};
class Faculty : public Person {                    // Derived class Faculty
private:
    string department;
public:
    Faculty(string n, int a, string dep) : Person(n, a), department(dep) {}
    string getDepartment() { return department; }
    void getDetails() override {
        Person::getDetails();
        cout << "Department: " << department << endl;
    }
};
int main() {
    Student s("Ravi", 21, 12345);
```

```
    s.setMajor("Computer Science");
    Faculty f("Amar", 50, "Engineering");
    cout << "Student Details:" << endl;
    s.getDetails();
    cout << endl;
    cout << "Faculty Details:" << endl;
    f.getDetails();
    cout << endl;
    return 0;
}
```

## Inherits from Person :-

Data members: department (string), employeeId (int)

Member functions:

setDepartment(string) to set the faculty member's department

getDepartment() to retrieve the department

Override getDetails() to include faculty-specific information

Derived Class (TeachingAssistant): (Multilevel Inheritance)

```
#include <iostream>
#include <string>
using namespace std;
class Person {                          // Base class
protected:
    string name;
    int age;
public:
    Person(string n, int a) : name(n), age(a) {}
```

```
    void setName(string n) { name = n; }
    void setAge(int a) { age = a; }
    string getName() { return name; }
    int getAge() { return age; }
    virtual void getDetails() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
class Student : public Person {                    // Derived class Student
private:
    int studentId;
    string major;
public:
    Student(string n, int a, int sid) : Person(n, a), studentId(sid) {}
    void setMajor(string m) { major = m; }
    string getMajor() { return major; }
    void getDetails() override {
        Person::getDetails();
        cout << "Student ID: " << studentId << endl;
        cout << "Major: " << major << endl;
    }
};
class Faculty : public Person {                    // Derived class Faculty
private:
    string department;
public:
    Faculty(string n, int a, string dep) : Person(n, a), department(dep) {}
```

```
    string getDepartment() { return department; }
    void getDetails() override {
        Person::getDetails();
        cout << "Department: " << department << endl;
    }
};
int main() {
    Student s("Amar", 20, 12345);
    s.setMajor("Computer Science");
    Faculty f("Dr. Vinay", 40, "Engineering");
    cout << "Student Details:" << endl;
    s.getDetails();
    cout << endl;
    cout << "Faculty Details:" << endl;
    f.getDetails();
    cout << endl;
    return 0;
}
```

## Inherits from Student (inherits indirectly from Person as well) :-

Data member: coursesTeaching (array/vector of strings)

Member functions:

setCoursesTeaching(string[]) to set the courses the TA is teaching

getCoursesTeaching() to retrieve the list of courses

Override getDetails() to include TA-specific information (e.g., courses)

Derived Class (ResearchAssistant): (Hierarchical Inheritance)

```
#include <iostream>
```

```
#include <string>
#include <vector>
using namespace std;
class Person {
protected:
	string name;
	int age;
public:
	Person(string n, int a) : name(n), age(a) {}
	void setName(string n) { name = n; }
	void setAge(int a) { age = a; }
	string getName() { return name; }
	int getAge() { return age; }
	virtual void getDetails() {
		cout << "Name: " << name << endl;
		cout << "Age: " << age << endl;
	}
};
class Student : public Person {                    // Derived class Student (inherits from Person)
protected:
	int studentId;
	string major;
public:
	Student(string n, int a, int sid, string m) : Person(n, a), studentId(sid), major(m) {}
	void setMajor(string m) { major = m; }
	string getMajor() { return major; }
	void getDetails() override {
		Person::getDetails();
```

```
        cout << "Student ID: " << studentId << endl;
        cout << "Major: " << major << endl;
    }
};
class TeachingAssistant : public Student {                    // Derived class TeachingAssistant
protected:
    vector<string> coursesTeaching;
public:
    TeachingAssistant(string n, int a, int sid, string m) : Student(n, a, sid, m) {}
    void setCoursesTeaching(const vector<string>& courses) { coursesTeaching = courses; }
    vector<string> getCoursesTeaching() { return coursesTeaching; }
    void getDetails() override {
        Student::getDetails();
        cout << "Courses Teaching:" << endl;
        for (const auto& course : coursesTeaching) {
            cout << "- " << course << endl;
        }
    }
};
class ResearchAssistant : public Student {                    // Derived class ResearchAssistant (inherits from Student)
protected:
    string researchTopic;
public:
    ResearchAssistant(string n, int a, int sid, string m, string topic) : Student(n, a, sid, m), researchTopic(topic) {}
    string getResearchTopic() { return researchTopic; }
    void getDetails() override {
```

```
        Student::getDetails();
        cout << "Research Topic: " << researchTopic << endl;
    }
};
int main() {
    vector<string> courses = {"Introduction to Programming", "Data Structures", "Algorithms"};
    TeachingAssistant ta("Arjun", 26, 12345, "Computer Science");
    ta.setCoursesTeaching(courses);
    ResearchAssistant ra("Bob", 27, 54321, "Physics", "Quantum Mechanics");
    cout << "Teaching Assistant Details:" << endl;
    ta.getDetails();
    cout << endl;
    cout << "Research Assistant Details:" << endl;
    ra.getDetails();
    cout << endl;
    return 0;
}
```

## Inherits from Person (separate inheritance from Student) :-

Data members: researchArea (string), supervisor (string)

Member functions:

setResearchArea(string) to set the research area

getResearchArea() to retrieve the research area

setSupervisor(string) to set the research supervisor

getSupervisor() to retrieve the supervisor

Override getDetails() to include RA-specific information

Derived Class (GraduateStudentTA): (Hybrid Inheritance)

```
#include <iostream>
#include <string>
class Person {
protected:
        std::string name;
        int age;

public:
      Person(const std::string& n, int a) : name(n), age(a) {}
        virtual void getDetails() const {
                std::cout << "Name: " << name << ", Age: " << age << std::endl;
        }
  virtual ~Person() {}
};
class Student {
protected:
        int studentId;
        std::string major;
public:
      Student(int id, const std::string& m) : studentId(id), major(m) {}
    void setMajor(const std::string& m) {
                major = m;
        }
  std::string getMajor() const {
                return major;
        }
    virtual void getStudentDetails() const {
                std::cout << "Student ID: " << studentId << ", Major: " << major << std::endl;
```

```
        }
    virtual ~Student() {}
};
class GraduateStudentTA : public Person, public Student {
private:
        std::string researchArea;
        std::string supervisor;
public:
        GraduateStudentTA(const std::string& n, int a, int id, const std::string& m)
                : Person(n, a), Student(id, m) {
void setResearchArea(const std::string& ra) {
                researchArea = ra;
        }
      std::string getResearchArea() const {
                return researchArea;
        }
void setSupervisor(const std::string& sup) {
                supervisor = sup;
        }
    std::string getSupervisor() const {
                return supervisor;
        }
void getDetails() const override {
                std::cout << "Graduate Student TA Details - ";
                Person::getDetails();
                std::cout << "Student Details - ";
                getStudentDetails();
                std::cout << "Research Area: " << researchArea << ", Supervisor: " << supervisor << std::endl;
```

```
    }
};
int main() {
    GraduateStudentTA gsta("Emma", 26, 12345, "Computer Science");
    gsta.setResearchArea("Machine Learning");
    gsta.setSupervisor("Dr. Johnson");
    gsta.getDetails();
  return 0;
}
```

Inherits from both Student and TeachingAssistant (combines functionality)

Might have additional data members or functions specific to graduate student TAs

```
#include <iostream>
#include <string.h>
using namespace std;
class TeachingAssistant {
protected:
    std::string jobTitle;
    std::string course;
public:
    TeachingAssistant(const std::string& title, const std::string& c)
        : jobTitle(title), course(c) {}
    virtual ~TeachingAssistant() {}
    void setJobTitle(const std::string& title) {
        jobTitle = title;
    }
    std::string getJobTitle() const {
```

```
        return jobTitle;
    }
    void setCourse(const std::string& c) {
        course = c;
    }
    std::string getCourse() const {
        return course;
    }
    virtual void getTeachingDetails() const {
        std::cout << "Job Title: " << jobTitle << ", Course: " << course << std::endl;
    }
};
class GraduateStudentTA : public person, public Student, public TeachingAssistant {
private:
    std::string researchArea;
    std::string supervisor;
public:
    GraduateStudentTA(const std::string& n, int a, int id, const std::string& m, const std::string& title,
const std::string& c)
        : Person(n, a), Student(id, m), TeachingAssistant(title, c) {}
    virtual ~GraduateStudentTA() {}
    void setResearchArea(const std::string& ra) {
        researchArea = ra;
    }
    std::string getResearchArea() const {
        return researchArea;
    }
    void setSupervisor(const std::string& sup) {
```

```
        supervisor = sup;
    }
    std::string getSupervisor() const {
        return supervisor;
    }
    void getDetails() const override {
        std::cout << "Graduate Student TA Details - ";
        Person::getDetails();
        std::cout << "Student Details - ";
        getStudentDetails();
        std::cout << "Teaching Assistant Details - ";
        getTeachingDetails();
        std::cout << "Research Area: " << researchArea << ", Supervisor: " << supervisor << std::endl;
    }
};
int main() {
    GraduateStudentTA gsta("Emma", 26, 12345, "Computer Science", "Teaching Assistant",
"Computer Science 101");
    gsta.setResearchArea("Machine Learning");
    gsta.setSupervisor("Dr. Johnson");
    gsta.getDetails();
    return 0;
}
```