**Q1. Define two classes namely student class having data members rollno, name, age and result class having data members mark1, mark2, mark3 derived from student implement these classes through main () showing the mechanism of single inheritance.**

#include <iostream>

using namespace std;

// Base class

class Student

{

protected:

    int rollno;

    string name;

    int age;

public:

    void getStudentData()

    {

        cout << "Enter Roll No: ";

        cin >> rollno;

        cin.ignore();  // to clear input buffer

        cout << "Enter Name: ";

        getline(cin, name);

        cout << "Enter Age: ";

        cin >> age;

    }

    void displayStudentData()

    {

        cout << "\nRoll No: " << rollno << endl;

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

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

    }

};

// Derived classclass Result : public Student

{

private:

    int mark1, mark2, mark3;

public:

    void getMarks()

    {

        cout << "Enter Mark 1: ";

        cin >> mark1;

        cout << "Enter Mark 2: ";

        cin >> mark2;

        cout << "Enter Mark 3: ";

        cin >> mark3;

    }

    void displayResult()

    {

        int total = mark1 + mark2 + mark3;

        float average = total / 3.0;

        displayStudentData();

        cout << "Marks: " << mark1 << ", " << mark2 << ", " << mark3 << endl;

        cout << "Total: " << total << endl;

        cout << "Average: " << average << endl;

    }

};

// Main function

int main()

{

    Result r;

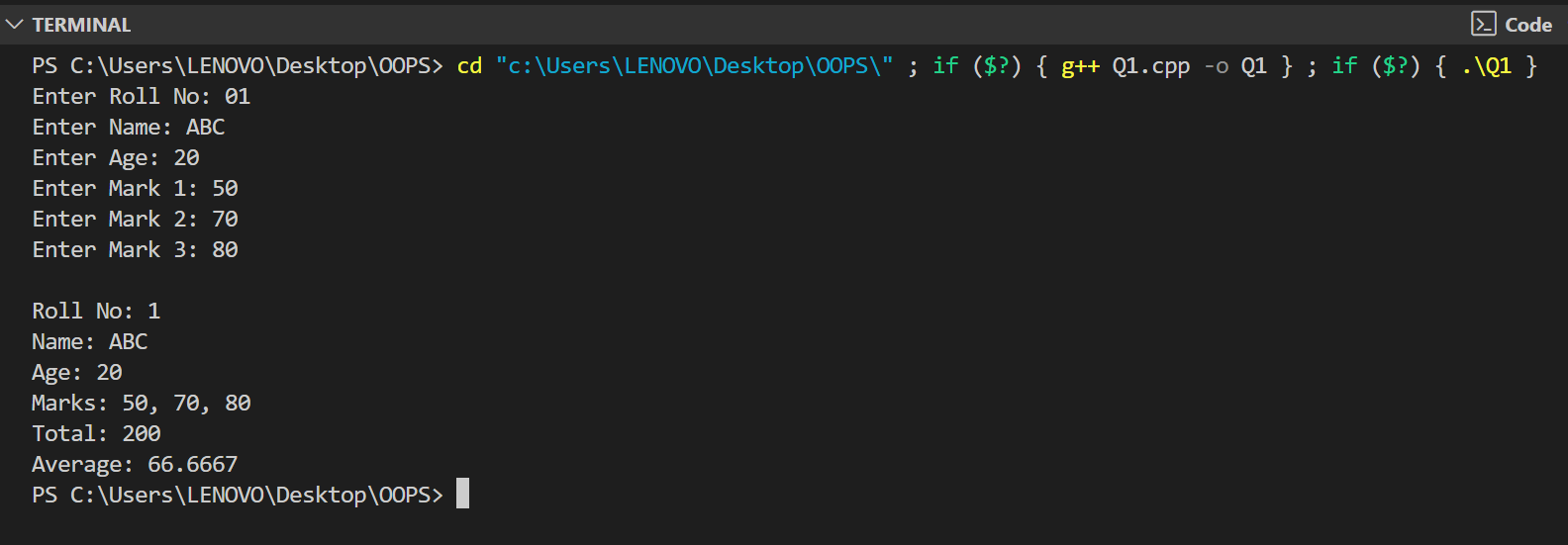
    r.getStudentData();

    r.getMarks();

    r.displayResult();

    return 0;

}



**Q2. Write a program for multiple inheritances. Define a class publisher that stores the name of the title and another class for sales detail, which stores the number of sales. Derive class book, which inherit both publisher and sales. Define function in the appropriate classes to get and print the details.**

#include <iostream>

using namespace std;

// Publisher class

class Publisher

{

protected:

    string title;

public:

    void getPublisherData()

    {

       // cout << "Enter Title Name: ";

        getline(cin, title);  // Properly takes full line input

    }

    void displayPublisherData()

    {

        cout << "Title: " << title << endl;

    }

};

// Sales class

class Sales

{

protected:

    int numberOfSales;

public:

    void getSalesData()

    {

        cout << "Enter Number of Sales: ";

        cin >> numberOfSales;

    }

    void displaySalesData()

    {

        cout << "Number of Sales: " << numberOfSales << endl;

    }

};

// Book class using multiple inheritance

class Book : public Publisher, public Sales

{

public:

    void getBookData()

    {

        getPublisherData();   // Input title first

        getSalesData();       // Then input sales

    }

    void displayBookData()

    {

        cout << "\n--- Book Details ---" << endl;

        displayPublisherData();

        displaySalesData();

    }

};

// Main function

int main()

{

    Book b;

    // Fix: Clear input buffer before getline

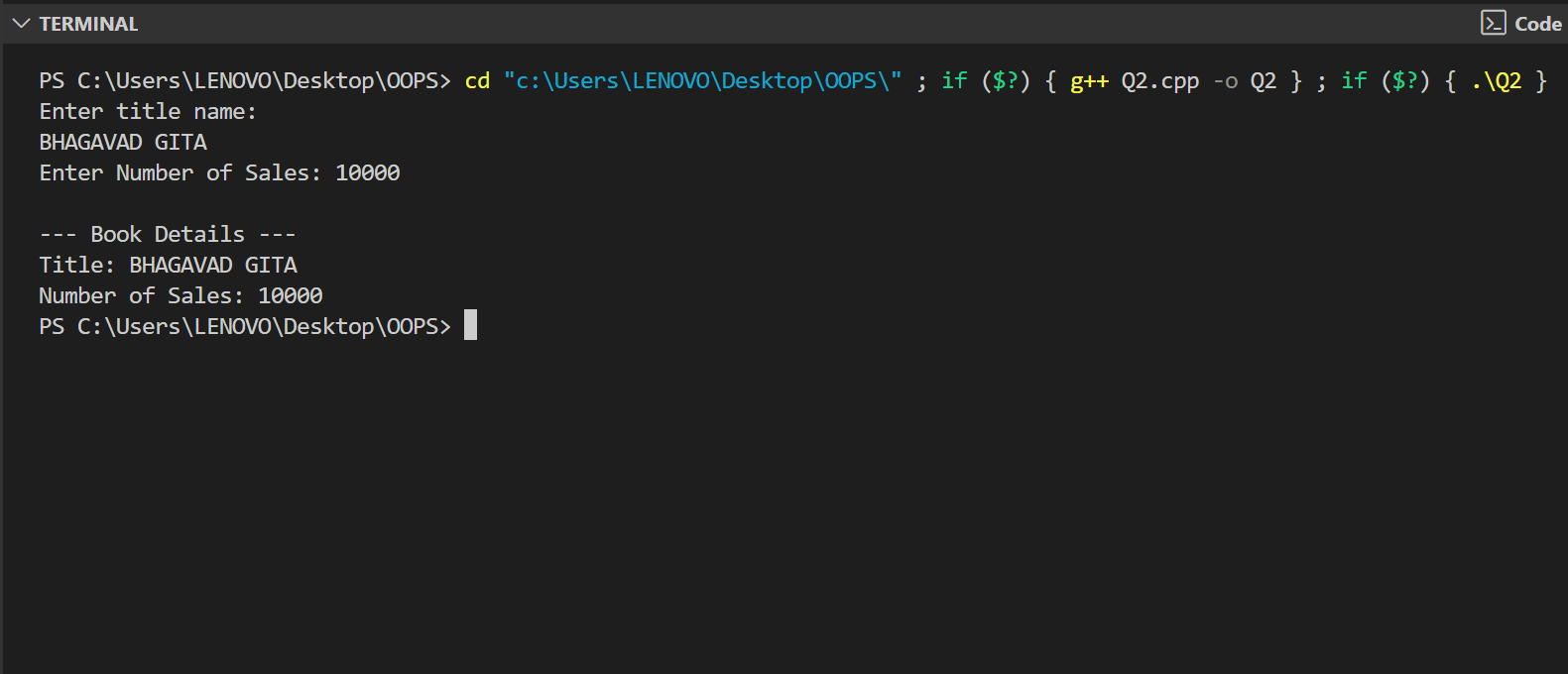
    cout << "Enter title name:\n";

    b.getBookData();

    b.displayBookData();

    return 0;

}

****

**Q3. Write a program that contains student as a base class, from which the three classes are arts, science and commerce have been derived illustrates the hierarchical inheritance with constructor. Display the student details along with their subject names.**

#include <iostream>

using namespace std;

// Base class

class Student

{

protected:

    string name;

    int rollNo;

public:

    // Constructor

    Student(string n, int r)

    {

        name = n;

        rollNo = r;

    }

    // Display common student details

    void displayStudent()

    {

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

        cout << "Roll No: " << rollNo << endl;

    }

};

// Derived class - Arts

class Arts : public Student

{

public:

    // Constructor

    Arts(string n, int r) : Student(n, r) {}

    // Display subject

    void displaySubjects()

    {

        displayStudent();

        cout << "Subjects: History, Political Science, Sociology" << endl;

    }

};

// Derived class - Science

class Science : public Student

{

public:

    // Constructor

    Science(string n, int r) : Student(n, r) {}

    void displaySubjects()

    {

        displayStudent();

        cout << "Subjects: Physics, Chemistry, Mathematics" << endl;

    }

};

// Derived class - Commerce

class Commerce : public Student

{

public:

    // Constructor

    Commerce(string n, int r) : Student(n, r) {}

    void displaySubjects()

    {

        displayStudent();

        cout << "Subjects: Accounting, Economics, Business Studies" << endl;

    }

};

// Main function

int main()

{

    // Creating objects of each derived class

    Arts a("Raj", 101);

    Science s("Priya", 102);

    Commerce c("Amit", 103);

    cout << "\n--- Arts Student ---\n";

    a.displaySubjects();

    cout << "\n--- Science Student ---\n";

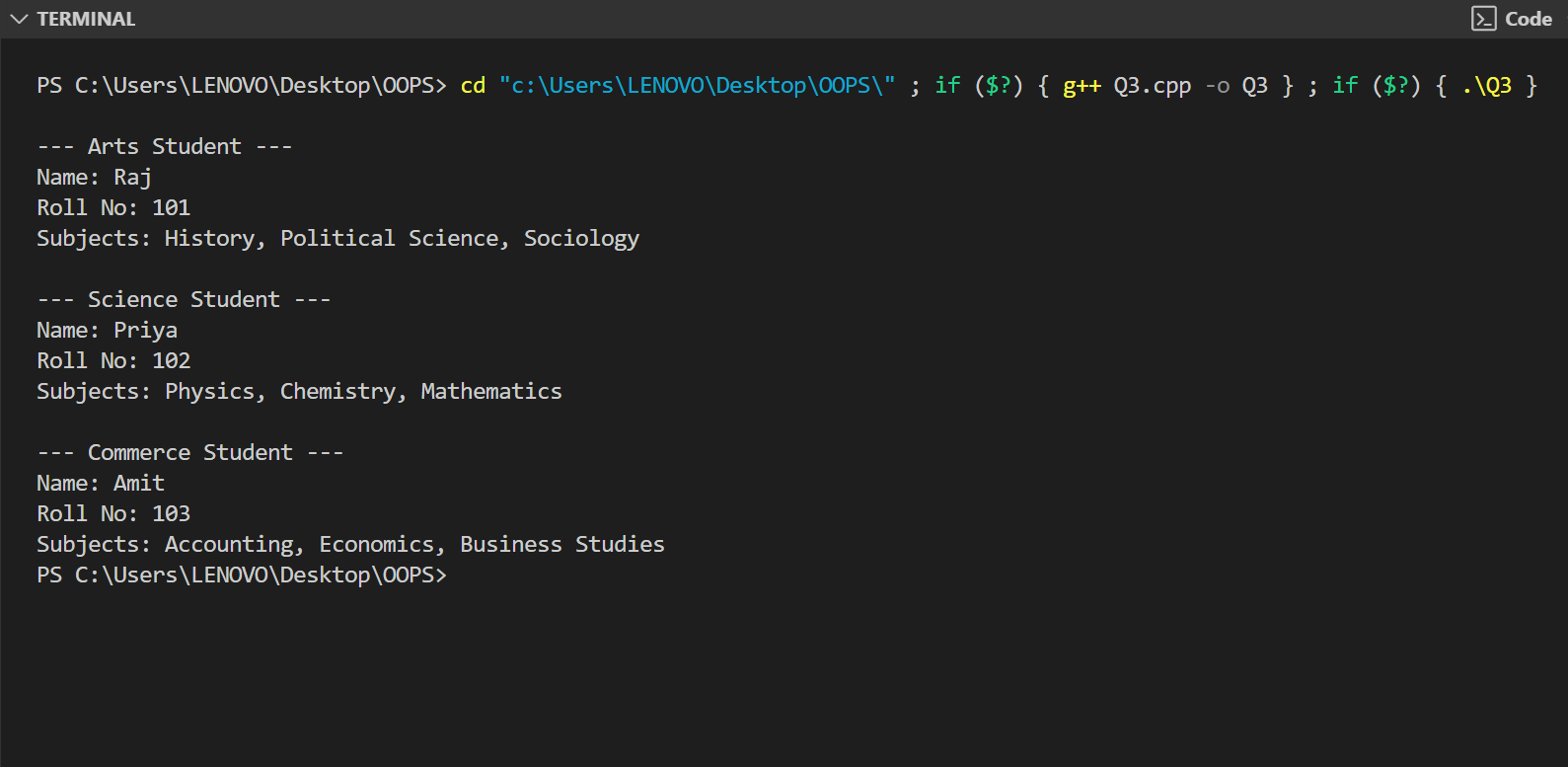
    s.displaySubjects();

    cout << "\n--- Commerce Student ---\n";

    c.displaySubjects()

    return 0;

}



**Q4. Write a C++ program to design a base class Person (name, address, phone\_no). Derive a class Employee (eno, ename) from Person. Derive a class Manager (Designation, Department name, basic salary) from Employee. Write a Menu Driven Program to:**

**a) Accept all details of ‘n’ Managers.**

**b) display manager having highest salary.**

#include <iostream>

#include <string>

using namespace std;

class Person

{

protected:

    string name;

    string address;

    string phone\_no;

public:

    void getPersonDetails()

    {

        cout << "Enter Name: ";

        getline(cin, name);

        cout << "Enter Address: ";

        getline(cin, address);

        cout << "Enter Phone Number: ";

        getline(cin, phone\_no);

    }

    void displayPersonDetails()

    {

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

        cout << "Address: " << address << endl;

        cout << "Phone Number: " << phone\_no << endl;

    }

};

class Employee : public Person

{

protected:

    int eno;

    string ename;

public:

    void getEmployeeDetails()

    {

        cout << "Enter Employee Number: ";

        cin >> eno;

        cin.ignore(); // To consume newline

        cout << "Enter Employee Name: ";

        getline(cin, ename);

        getPersonDetails();

    }

    void displayEmployeeDetails()

    {

        cout << "Employee No: " << eno << endl;

        cout << "Employee Name: " << ename << endl;

        displayPersonDetails();

    }

};

class Manager : public Employee

{

private:

    string designation;

    string department;

    float basic\_salary;

public:

    void getManagerDetails()

    {

        getEmployeeDetails();

        cout << "Enter Designation: ";

        getline(cin, designation);

        cout << "Enter Department Name: ";

        getline(cin, department);

        cout << "Enter Basic Salary: ";

        cin >> basic\_salary;

        cin.ignore();

    }

    void displayManagerDetails()

    {

        displayEmployeeDetails();

        cout << "Designation: " << designation << endl;

        cout << "Department: " << department << endl;

        cout << "Basic Salary: " << basic\_salary << endl;

        cout << "--------------------------\n";

    }

    float getSalary()

    {

        return basic\_salary;

    }

};

int main()

{

    int n, choice;

    cout << "Enter number of managers: ";

    cin >> n;

    cin.ignore();

    Manager managers[100]; // Assuming max 100 managers

    bool running = true;

    while (running)

    {

        cout << "\n--- MENU ---\n";

        cout << "1. Accept Details of Managers\n";

        cout << "2. Display Manager with Highest Salary\n";

        cout << "3. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        cin.ignore();

        switch (choice)

        {

        case 1:

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

            {

                cout << "\nEnter details for Manager " << i + 1 << ":\n";

                managers[i].getManagerDetails();

            }

            break;

        case 2:

        {

            if (n == 0)

            {

                cout << "No manager data entered.\n";

                break;

            }

            float maxSalary = managers[0].getSalary();

            int index = 0;

            for (int i = 1; i < n; i++)

            {

                if (managers[i].getSalary() > maxSalary)

                {

                    maxSalary = managers[i].getSalary();

                    index = i;

                }

            }

            cout << "\nManager with Highest Salary:\n";

            managers[index].displayManagerDetails();

            break;

        }

        case 3:

            running = false;

            break;

        default:

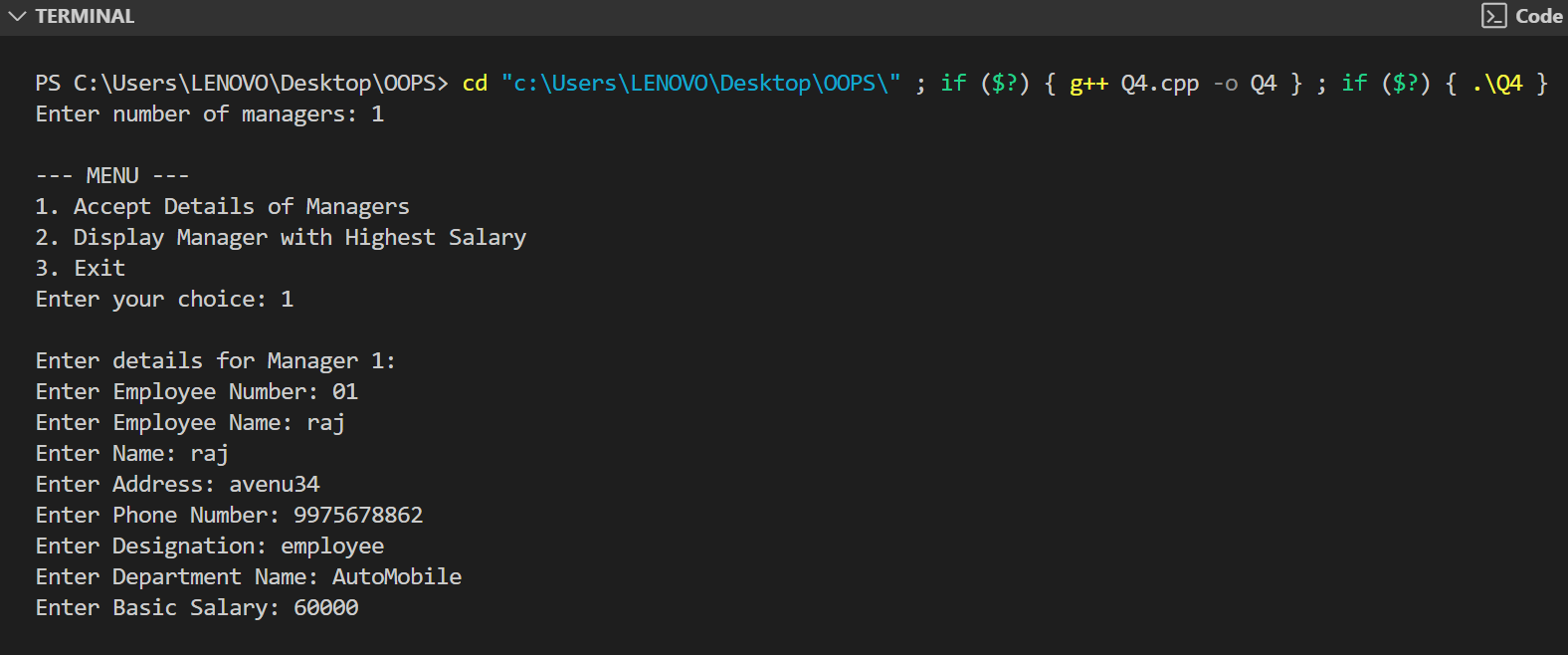
            cout << "Invalid choice. Try again.\n";

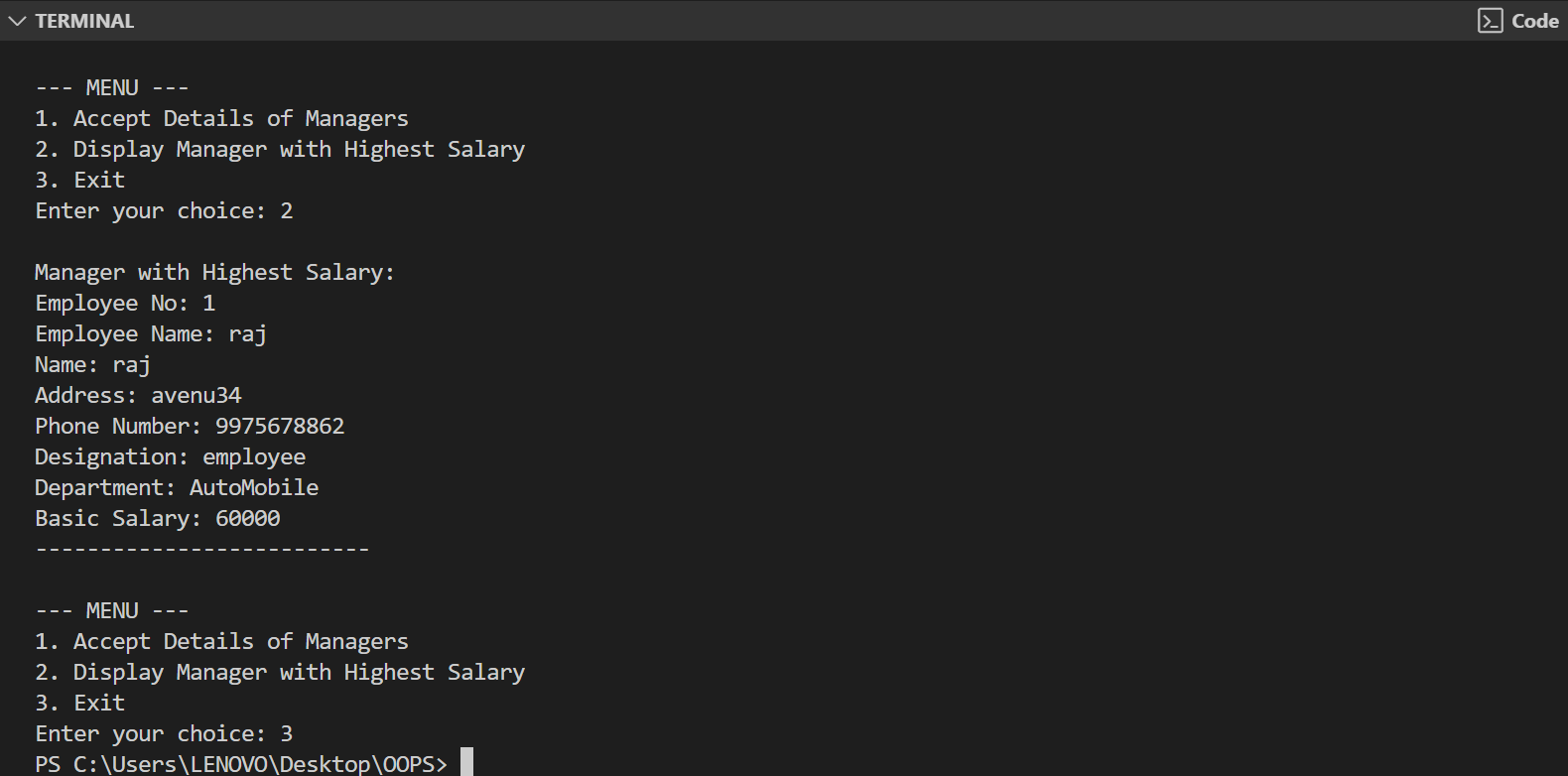
        }

    }

    return 0;

}





**Q5. Write a C++ program to use pointer for both base and derived classes and call the member function. Use Virtual keyword**

#include <iostream>

using namespace std;

// Base class

class Base

{

public:

    virtual void show()

    {

        cout << "Base class show() function called." << endl;

    }

};

// Derived class

class Derived : public Base

{

public:

    void show() override

    {

        cout << "Derived class show() function called." << endl;

    }

};

int main()

{

    Base\* basePtr;         // Pointer of base class

    Base baseObj;          // Object of base class

    Derived derivedObj;    // Object of derived class

    // Pointing to base class object

    basePtr = &baseObj;

    basePtr->show();       // Calls Base class version

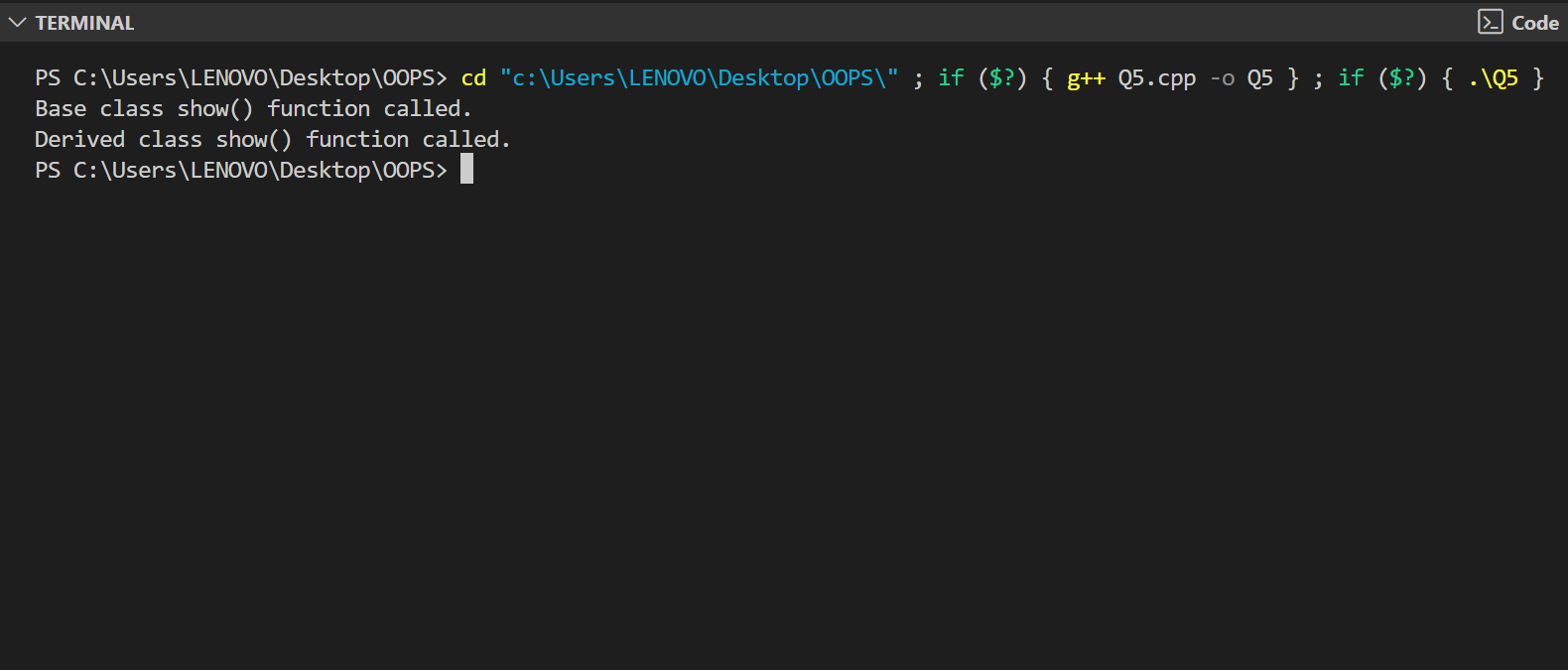
    // Pointing to derived class object

    basePtr = &derivedObj;

    basePtr->show();       // Calls Derived class version due to virtual function

    return 0;

}

****

**Q6. Define a class Weight having data members Kg and Gram and constructor for accepting weights and a member function to display the weights. Your program must also have a capability so that if you enter a weight in from of total grams then it converts these grams to equivalent kg and remaining gram and stores it in respective data members.**

#include <iostream>

using namespace std;

class Weight {

private:

    int kg;

    int gram;

public:

    // Constructor that accepts kg and gram

    Weight(int k, int g) {

        int totalGrams = k \* 1000 + g;

        kg = totalGrams / 1000;

        gram = totalGrams % 1000;

    }

    // Static method to create object from total grams

    static Weight fromGrams(int totalGrams) {

        int k = totalGrams / 1000;

        int g = totalGrams % 1000;

        return Weight(k, g);

    }

    void display() {

        cout << "Weight: " << kg << " kg and " << gram << " gram" << endl;

    }

};

int main() {

    int choice;

    cout << "1. Enter weight as Kg and Gram\n";

    cout << "2. Enter weight as total Grams\n";

    cout << "Enter your choice: ";

    cin >> choice;

    if (choice == 1) {

        int kg, g;

        cout << "Enter Kilograms: ";

        cin >> kg;

        cout << "Enter Grams: ";

        cin >> g;

        Weight w(kg, g);

        w.display();

    } else if (choice == 2) {

        int totalGrams;

        cout << "Enter total grams: ";

        cin >> totalGrams;

        Weight w = Weight::fromGrams(totalGrams);

        w.display();

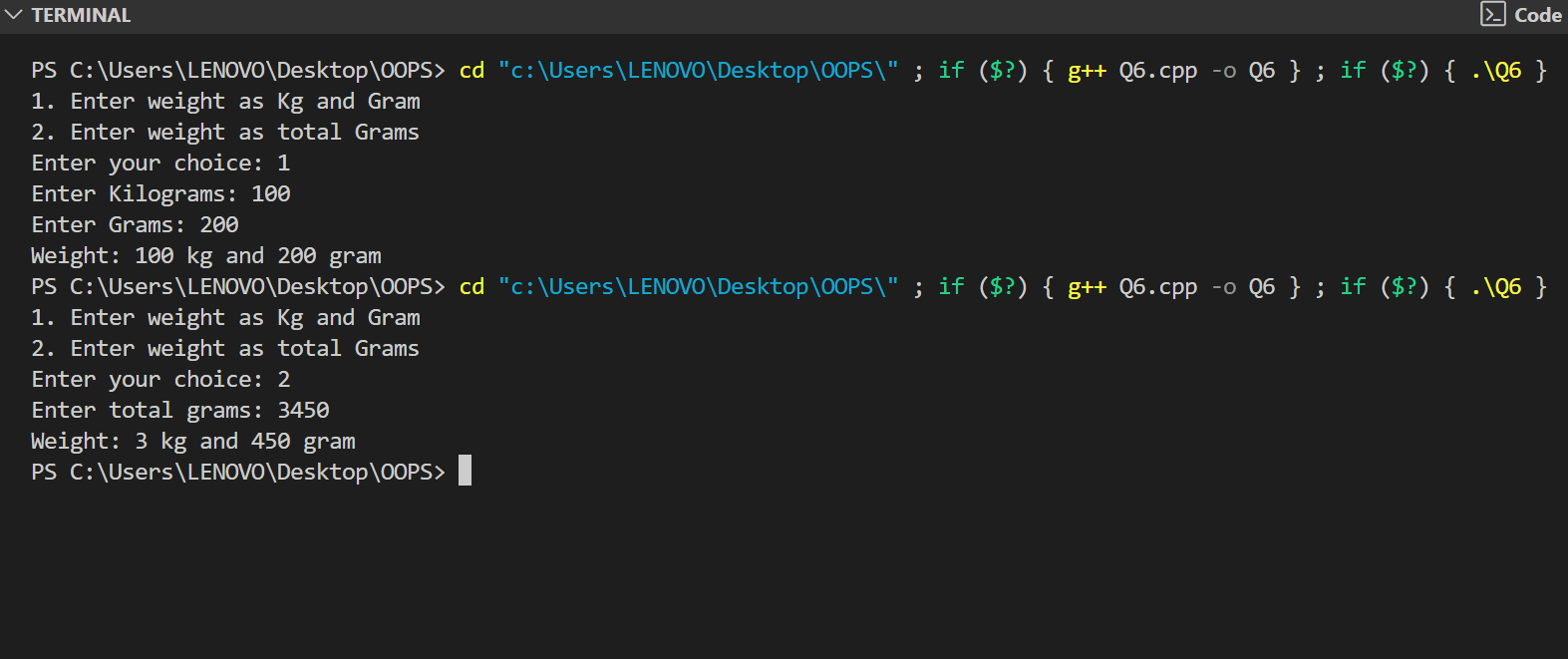
    } else {

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

    }

    return 0;

}



**Q7. Write a C++ program to illustrate the concept of Pointer to Objects.**

#include <iostream>

using namespace std;

class Student

{

private:

    int rollNo;

    string name;

public:

    void getData()

    {

        cout << "Enter Roll Number: ";

        cin >> rollNo;

        cin.ignore(); // Clear newline from buffer

        cout << "Enter Name: ";

        getline(cin, name);

    }

    void displayData()

    {

        cout << "Roll Number: " << rollNo << endl;

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

    }

};

int main()

{

    Student s1;          // Create object

    Student\* ptr;        // Declare pointer to object

    ptr = &s1;           // Assign address of object to pointer

    cout << "Enter student details\n";

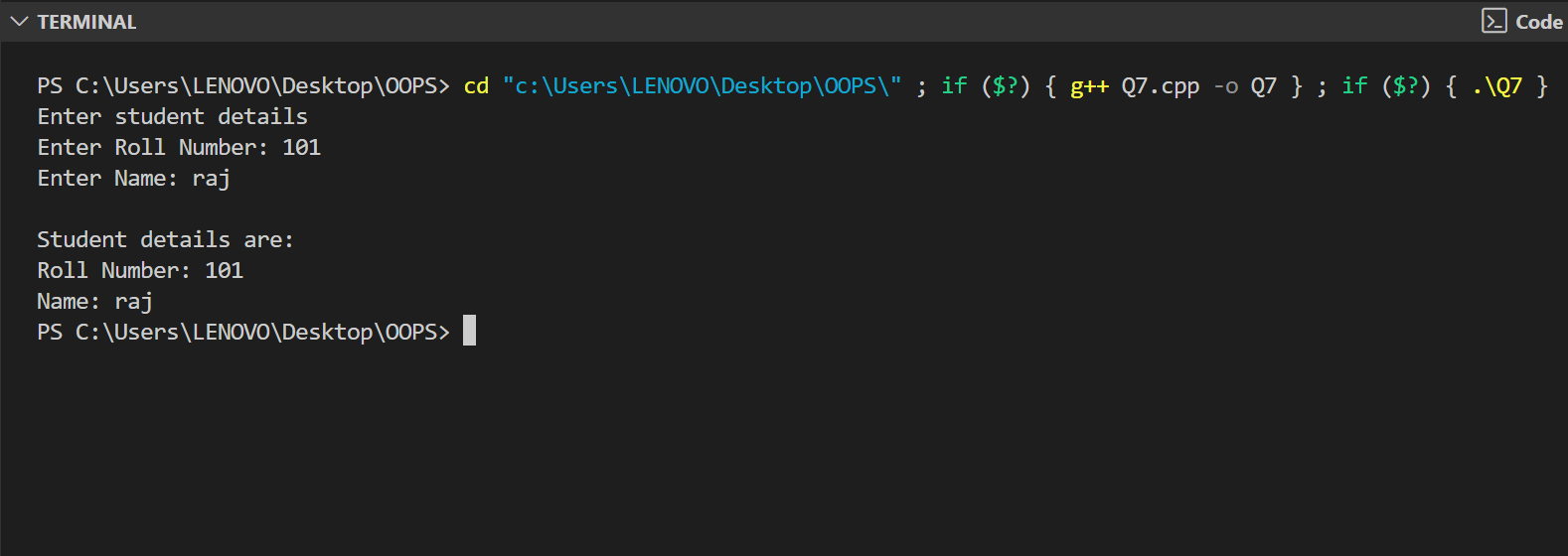
    ptr->getData();      // Access using pointer

    cout << "\nStudent details are:\n";

    ptr->displayData();  // Access using pointer

    return 0;

}



**Q8. Write a program to overload the operator ‘\*’ .**

#include <iostream>

using namespace std;

class Number

{

private:

    int value;

public:

    // Constructor

    Number(int v = 0)

    {

        value = v;

    }

    // Overloading '\*' operator

    Number operator\*(const Number& obj)

    {

        Number result;

        result.value = this->value \* obj.value;

        return result;

    }

    void display()

    {

        cout << "Value: " << value << endl;

    }

};

int main()

{

    int a, b;

    cout << "Enter two numbers: ";

    cin >> a >> b;

    Number num1(a), num2(b);

    Number result;

    result = num1 \* num2;  // Using overloaded '\*' operator

    cout << "Result of multiplication ";

    result.display();

    return 0;

}

