**ASSIGNMENT-7 (OOPS). INHERITANCE**

**Question 1-3: Write a program to check the difference in outputs for different modes of inheritance. Consider a class named Base1 with different type of data variables and member functions, i.e. private, public and protected. Inherit this class in another class named derived, which too has different type of data variables and member functions, i.e. private, public and protected. Run this program for private, public and protected mode and display the output. The detailed statements are as follows:**  
**Q1: In Base1, declare a, b and c of type int as private, public and protected respectively. In derived class,**  
  
**declare d, e and f of type int as private, public and protected respectively. In Derived class, Also define a private member function named Sum in Derived class that finds the sum of a, b, c, d, e and f data members. Also define a pubic function average that computes the average using sum returned by sum function. Inherit base class in private mode to solve this and display average on console.**

INPUT:

#include <iostream>

using namespace std;

class Base1 {

private:

    int a;

public:

    int b;

protected:

    int c;

public:

    Base1(int a, int b, int c) : a(a), b(b), c(c) {}

    friend class Derived;

};

class Derived : private Base1 {

private:

    int d;

public:

    int e;

protected:

    int f;

private:

    int Sum() {

        return a + b + c + d + e + f;

    }

public:

    Derived(int a, int b, int c, int d, int e, int f)

        : Base1(a, b, c), d(d), e(e), f(f) {}

    float Average() {

        return Sum() / 6.0;

    }

};

int main() {

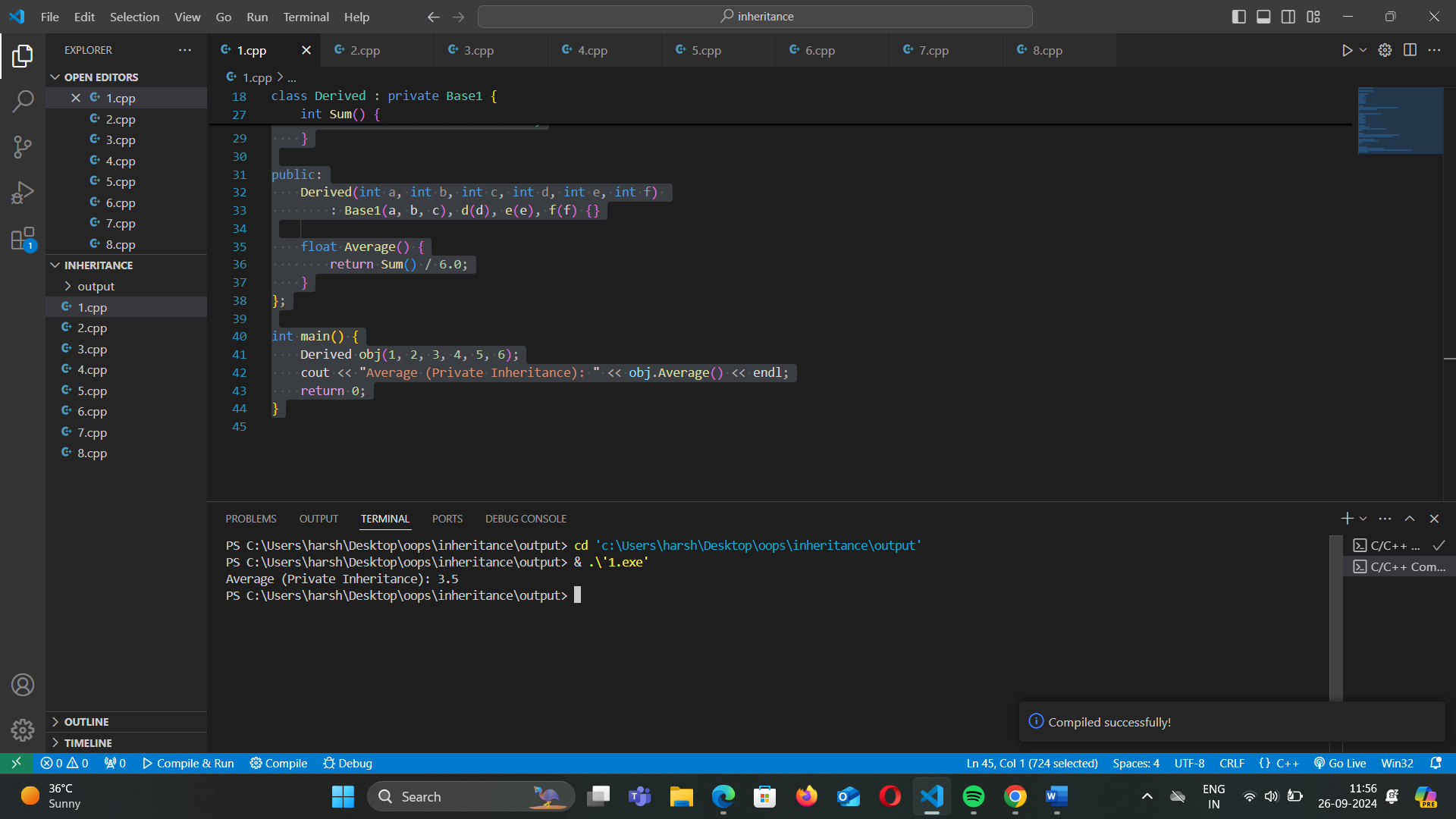
    Derived obj(1, 2, 3, 4, 5, 6);

    cout << "Average (Private Inheritance): " << obj.Average() << endl;

    return 0;

}

OUTPUT:



### **Explanation of Key Concepts:**

1. **Private Inheritance**:
   1. When a class is inherited privately, all public and protected members of the base class become private members in the derived class.
   2. Private members of the base class are not accessible directly in the derived class.
2. **Base Class** (Base1):
   1. **Private Members**: a (only accessible within Base1).
   2. **Public Members**: b (accessible outside the class when publicly inherited).
   3. **Protected Members**: c (accessible in derived classes when inherited).
3. **Derived Class** (Derived):
   1. **Private Members**: d, and a private member function Sum().
   2. **Public Members**: e and a public function Average().
   3. **Protected Members**: f.
4. **Base Class** (Base1):
5. The class Base1 has three data members:
   1. a is **private**, meaning it can only be accessed within Base1 itself.
   2. b is **public**, meaning it can be accessed directly if inherited as public or protected.
   3. c is **protected**, meaning it can only be accessed within derived classes or friend functions.
6. We have a constructor to initialize these members and declared the Derived class as a friend, giving it access to private members like a.
7. **Derived Class** (Derived):
8. This class inherits from Base1 **privately**.
   1. Even though b is public in Base1, it becomes private in Derived because we are using private inheritance.
   2. Similarly, c, which is protected in Base1, also becomes private in Derived.
   3. Private inheritance means that all members of Base1 (a, b, c) are private in Derived, making them inaccessible to any code outside Derived.
9. **Private Function Sum()**:
10. This function calculates the sum of all the members from both the base class (a, b, c) and derived class (d, e, f).
11. It’s private, so it can’t be accessed from outside the Derived class directly, but it can be used internally within the Derived class.
12. **Public Function Average()**:
13. This function calculates the average using the result from the private Sum() function and is publicly accessible.
14. The sum is divided by 6.0 to get the average.

**QUES-2  In Base1, declare a, b and c of type int as private, public and protected respectively. In derived class,  declare d, e and f of type int as private, public and protected respectively. In Derived class, Also define a private member function named Sum in Derived class that finds the sum of a, b, c, d, e and f data members. Also define a pubic function average that computes the average using sum returned by sum function. Inherit base class in public mode to solve this and display average on console.**

INPUT:

#include <iostream>

using namespace std;

class Base1 {

protected:

    int a;

public:

    int b;

protected:

    int c;

public:

    Base1(int a, int b, int c) : a(a), b(b), c(c) {}

};

class Derived : public Base1 {

private:

    int d;

public:

    int e;

protected:

    int f;

private:

    int Sum() {

        return a + b + c + d + e + f;

    }

public:

    Derived(int a, int b, int c, int d, int e, int f)

        : Base1(a, b, c), d(d), e(e), f(f) {}

    float Average() {

        return Sum() / 6.0;

    }

};

int main() {

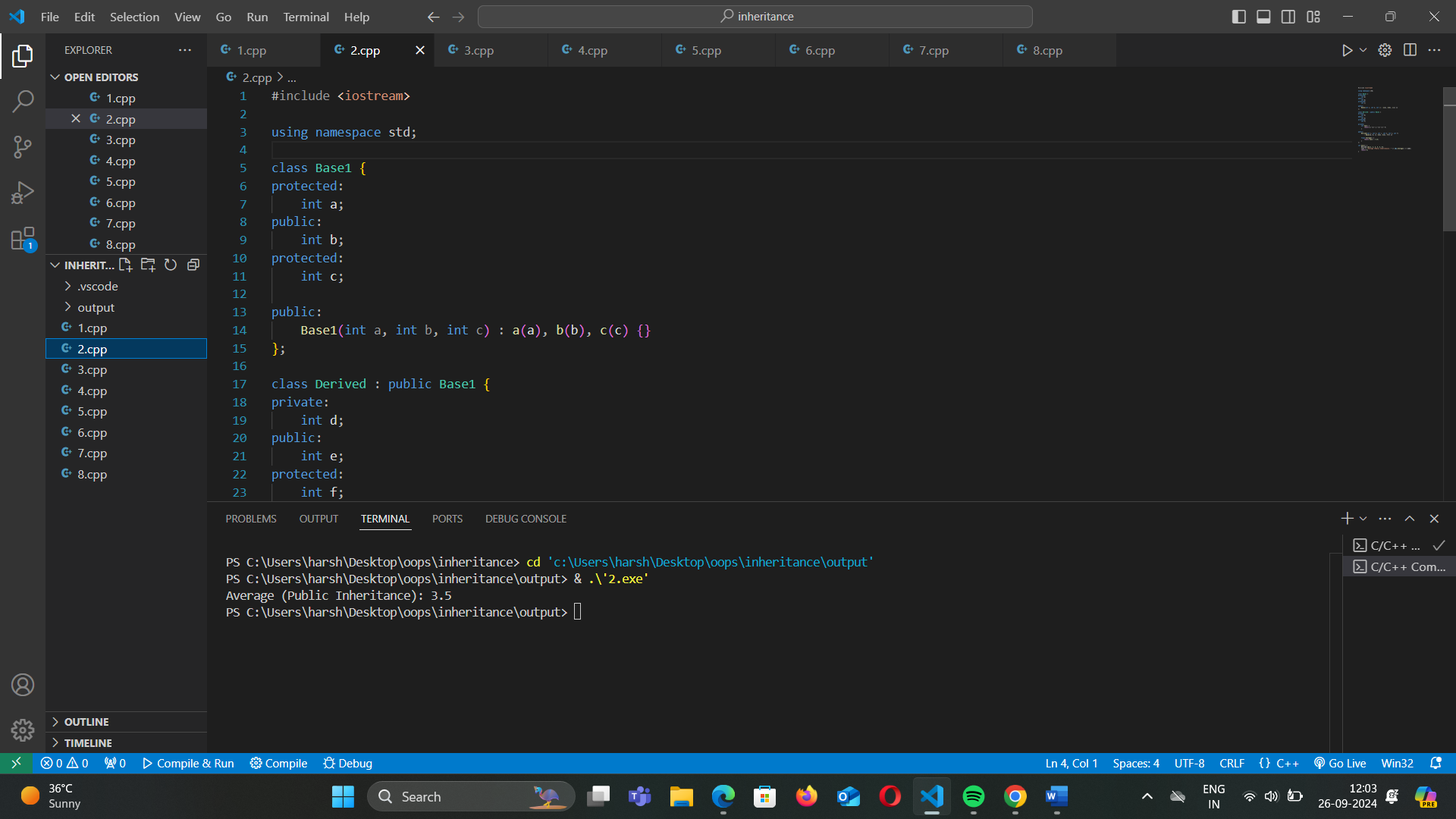
    Derived obj(1, 2, 3, 4, 5, 6);

    cout << "Average (Public Inheritance): " << obj.Average() << endl;

    return 0;

}

OUTPUT:



### **Explanation of Public Inheritance:**

In this program, we are using **public inheritance** to inherit the base class Base1 into the derived class Derived. The program demonstrates how data members from Base1 are accessible in Derived and how a sum and average are computed using both base and derived class members.

### **Key Concepts:**

1. **Public Inheritance**:
   1. When a class is inherited **publicly**, all public members of the base class remain public in the derived class, and protected members remain protected in the derived class.
   2. Private members of the base class are not accessible in the derived class.
2. **Base Class (Base1)**:
   1. **Protected Member**: a — accessible only within Base1 and its derived classes.
   2. **Public Member**: b — accessible to all, including outside the class.
   3. **Protected Member**: c — accessible only within Base1 and its derived classes.
3. **Derived Class (Derived)**:
   1. **Private Members**: d and a private member function Sum().
   2. **Public Members**: e and a public function Average().
   3. **Protected Members**: f.
4. **Base Class (Base1)**:
   1. The class Base1 has three data members:
      1. a is **protected**, which means it is accessible to derived classes, but not directly accessible outside Base1.
      2. b is **public**, which means it is accessible to everyone.
      3. c is **protected**, which again means it is accessible only in derived classes and Base1.
   2. The constructor initializes these values.
5. **Derived Class (Derived)**:
   1. This class inherits Base1 publicly, which means:
      1. b remains public.
      2. a and c remain protected, and are accessible inside the Derived class.
   2. The Sum() function calculates the sum of all six members (a, b, c from Base1 and d, e, f from Derived).
   3. The Average() function is public and returns the average by calling the Sum() function.
6. **Sum Function**:
   1. This is a private function that calculates the sum of the members from both the base class and derived class.
   2. The formula is: a + b + c + d + e + f.
7. **Average Function**:
   1. This is a public function that calculates the average by dividing the sum by 6.0.

**QUES-3  In Base1, declare a, b and c of type int as private, public and protected respectively. In derived class,  declare d, e and f of type int as private, public and protected respectively. In Derived class, Also define a private member function named Sum in Derived class that finds the sum of a, b, c, d, e and f data members. Also define a pubic function average that computes the average using sum returned by sum function. Inherit base class in protected mode to solve this and display average on console.**

INPUT:

#include <iostream>

using namespace std;

class Base1 {

protected:

    int a;

public:

    int b;

protected:

    int c;

public:

    Base1(int a, int b, int c) : a(a), b(b), c(c) {}

};

class Derived : protected Base1 {

private:

    int d;

public:

    int e;

protected:

    int f;

private:

    int Sum() {

        return a + b + c + d + e + f;

    }

public:

    Derived(int a, int b, int c, int d, int e, int f)

        : Base1(a, b, c), d(d), e(e), f(f) {}

    float Average() {

        return Sum() / 6.0;

    }

};

int main() {

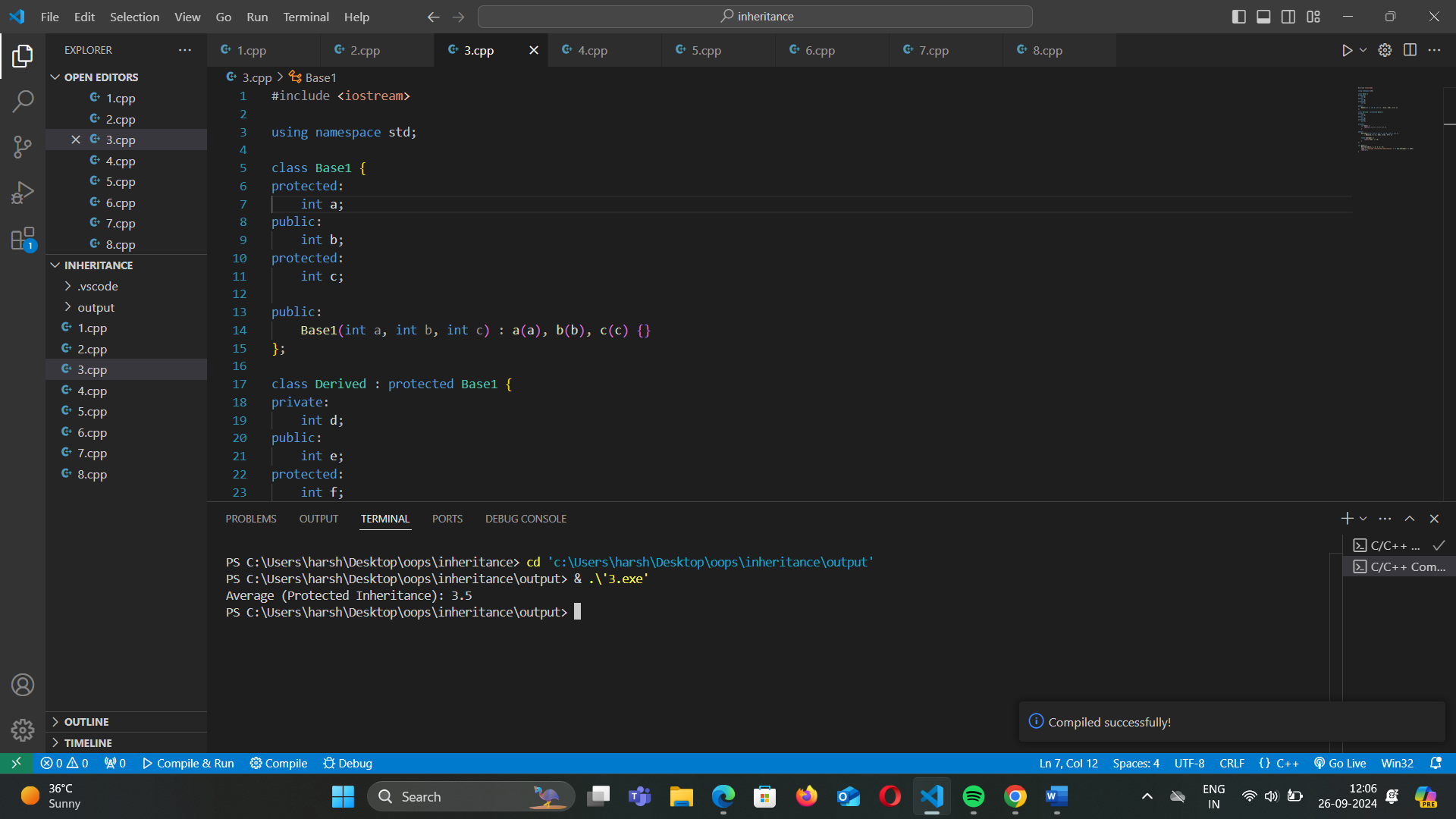
    Derived obj(1, 2, 3, 4, 5, 6);

    cout << "Average (Protected Inheritance): " << obj.Average() << endl;

    return 0;

}

OUTPUT:



### **Explanation of Protected Inheritance:**

In this example, the base class Base1 is inherited by the derived class Derived using **protected inheritance**. This means that all public and protected members of Base1 become **protected** in the derived class, while private members of the base class remain inaccessible directly.

### **Key Concepts:**

1. **Protected Inheritance**:
   1. When a class is inherited **protectedly**, the public and protected members of the base class become **protected** in the derived class.
   2. Private members of the base class are not accessible in the derived class.
2. **Base Class (Base1)**:
   1. **Protected Member**: a — accessible in derived classes but not outside the class.
   2. **Public Member**: b — becomes protected in the derived class.
   3. **Protected Member**: c — accessible in derived classes but not outside the class.
3. **Derived Class (Derived)**:
   1. **Private Members**: d and a private member function Sum().
   2. **Public Members**: e and a public function Average().
   3. **Protected Members**: f.
4. **Base Class (Base1)**:
   1. The base class Base1 has three data members:
      1. a is **protected** and can only be accessed within Base1 and its derived classes.
      2. b is **public**, but after protected inheritance, it becomes protected in the derived class.
      3. c is **protected** and is accessible in Base1 and derived classes.
5. **Derived Class (Derived)**:
   1. This class inherits Base1 using **protected inheritance**. As a result:
      1. b (public in Base1) becomes **protected** in Derived.
      2. a and c (protected in Base1) remain **protected** in Derived.
   2. The Sum() function calculates the sum of the six members (three from Base1 and three from Derived).
   3. The Average() function returns the average of the sum by dividing by 6.0.
6. **Sum Function**:
   1. This private function calculates the sum of the base class and derived class members:
      1. a + b + c + d + e + f.
7. **Average Function**:
   1. The public Average() function computes the average by calling the private Sum() function and dividing by 6.0.

QUES-4 Write a program to implement different types of inheritance.  
a) Single

INPUT:

#include <iostream>

using namespace std;

class Base {

public:

    void displayBase() {

        cout << "Base class method." << endl;

    }

};

class SingleDerived : public Base {

public:

    void displaySingle() {

        cout << "Single derived class method." << endl;

    }

};

int main() {

    SingleDerived sd;

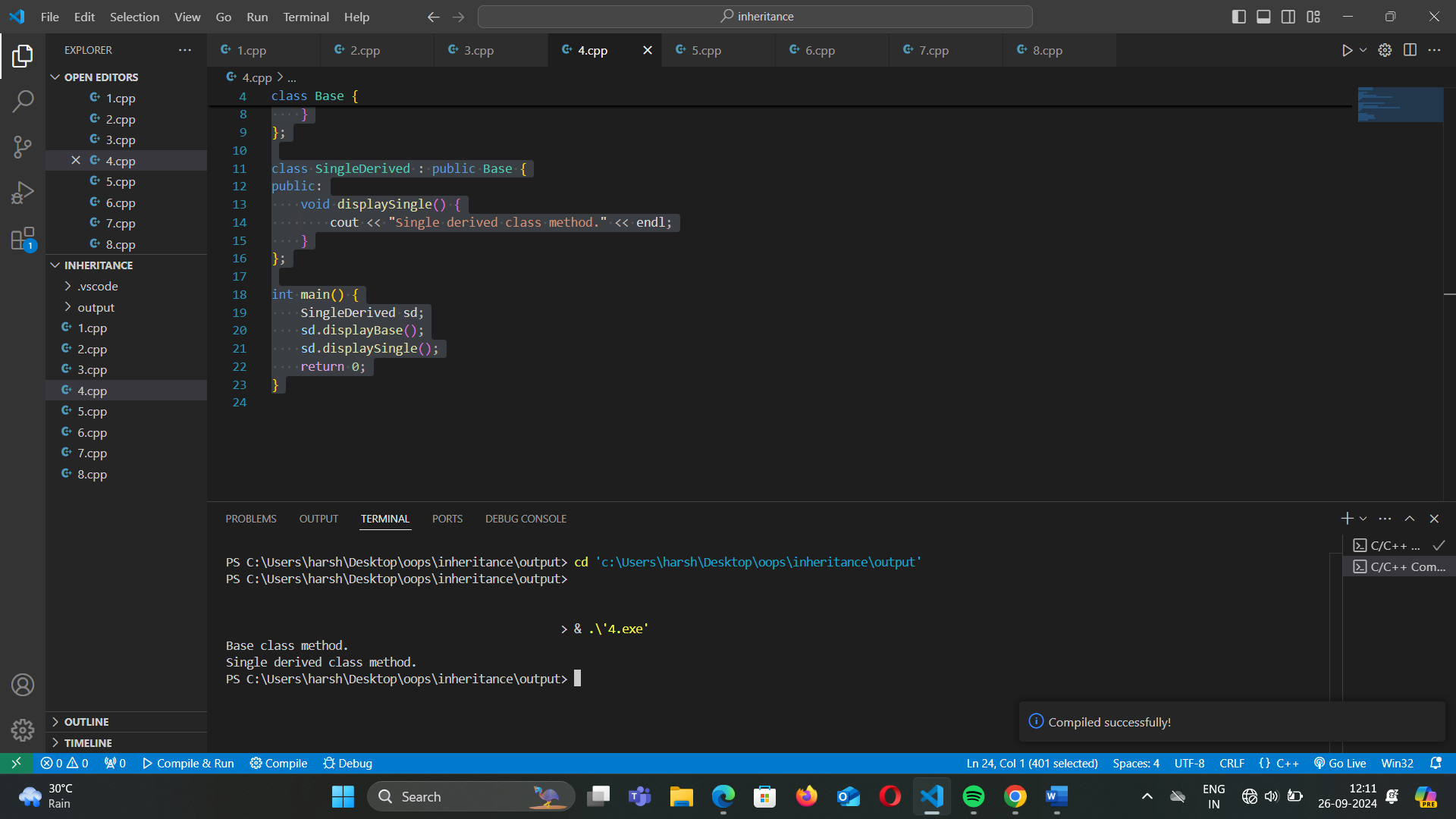
    sd.displayBase();

    sd.displaySingle();

    return 0;

}

OUTPUT:



QUES-5 b) Multiple

INPUT: #include <iostream>

using namespace std;

class ClassA {

public:

    void displayA() {

        cout << "Class A method." << endl;

    }

};

class ClassB {

public:

    void displayB() {

        cout << "Class B method." << endl;

    }

};

class MultipleDerived : public ClassA, public ClassB {

public:

    void displayMultiple() {

        cout << "Multiple derived class method." << endl;

    }

};

int main() {

    MultipleDerived md;

    md.displayA();

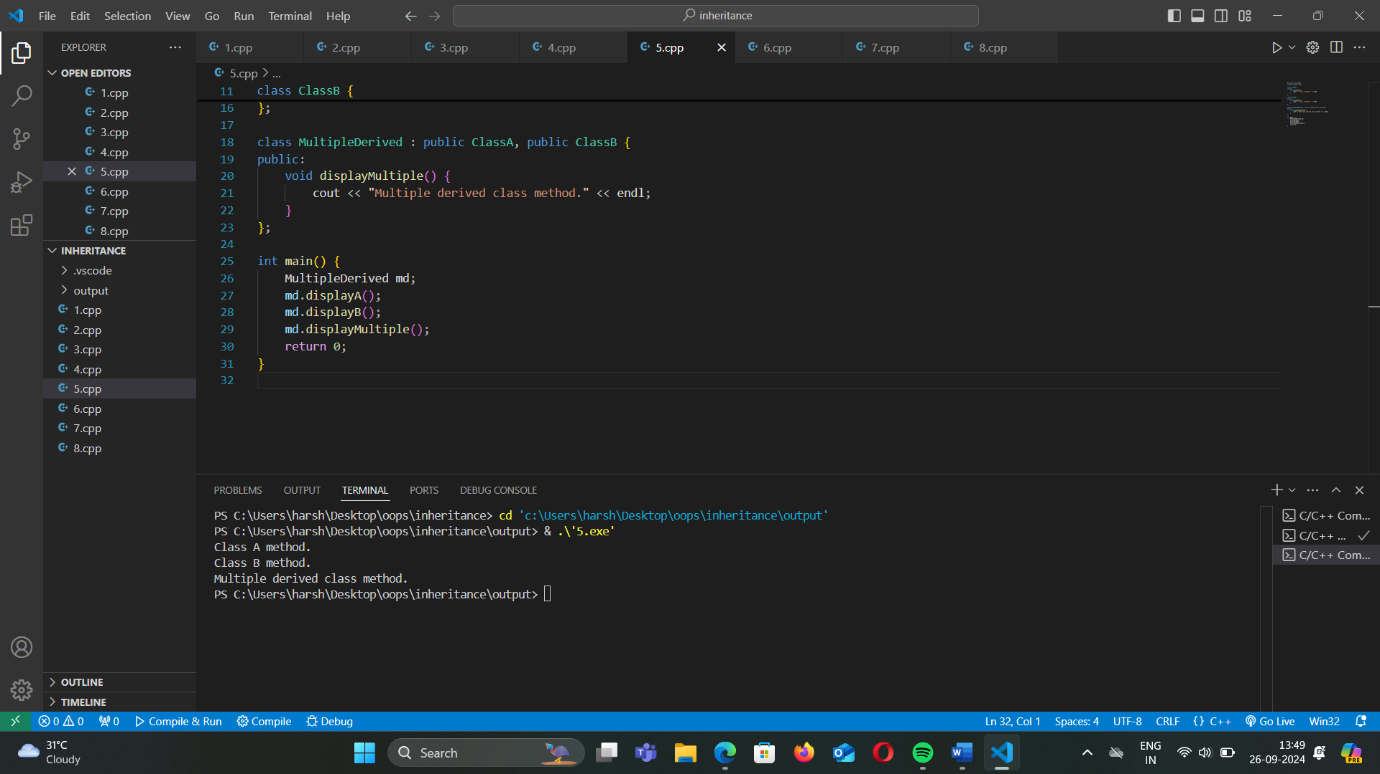
    md.displayB();

    md.displayMultiple();

    return 0;

}

OUTPUT:



QUES-6 c) Multi-level

INPUT: #include <iostream>

using namespace std;

class MultiBase {

public:

    void displayMultiBase() {

        cout << "Multi-level base class method." << endl;

    }

};

class MultiDerived1 : public MultiBase {

public:

    void displayMultiDerived1() {

        cout << "First derived class method." << endl;

    }

};

class MultiDerived2 : public MultiDerived1 {

public:

    void displayMultiDerived2() {

        cout << "Second derived class method." << endl;

    }

};

int main() {

    MultiDerived2 md2;

    md2.displayMultiBase();

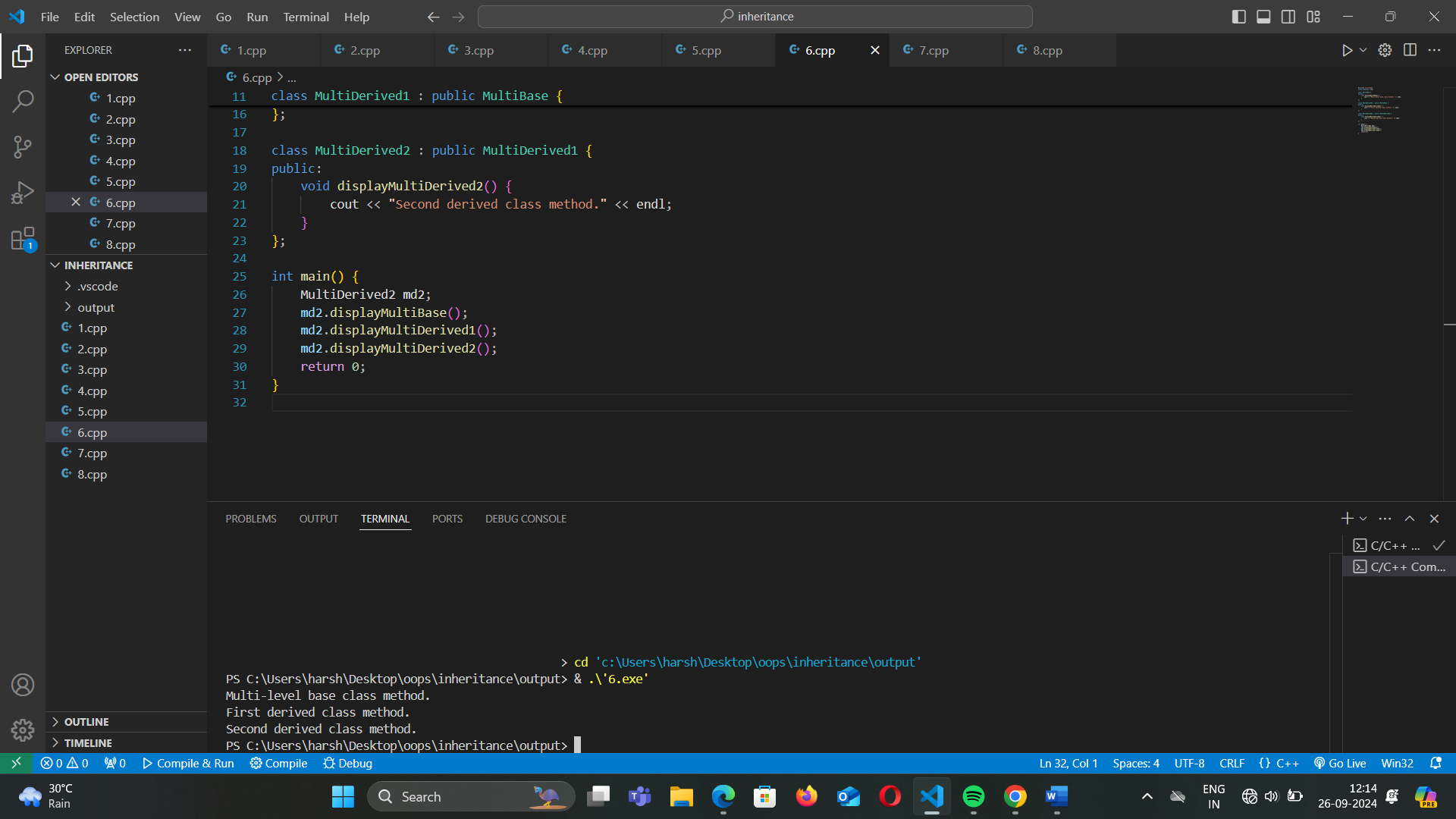
    md2.displayMultiDerived1();

    md2.displayMultiDerived2();

    return 0;

}

OUTPUT:



QUES-6 d) hybrid

INPUT: #include <iostream>

using namespace std;

class HybridBase1 {

public:

    void displayHybridBase1() {

        cout << "Hybrid Base1 class method." << endl;

    }

};

class HybridBase2 {

public:

    void displayHybridBase2() {

        cout << "Hybrid Base2 class method." << endl;

    }

};

class HybridDerived : public HybridBase1, public HybridBase2 {

public:

    void displayHybridDerived() {

        cout << "Hybrid derived class method." << endl;

    }

};

int main() {

    HybridDerived hd;

    hd.displayHybridBase1();

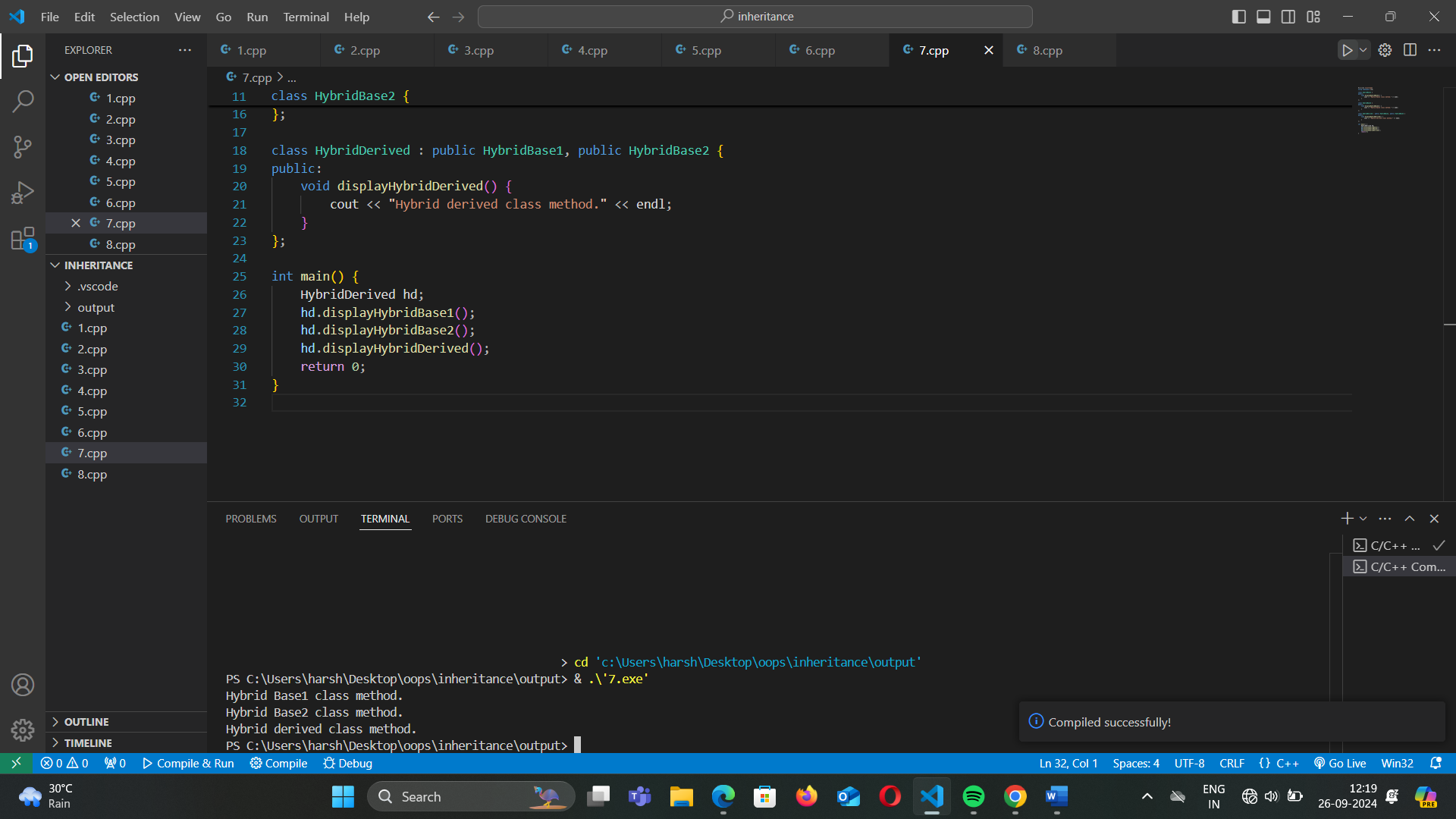
    hd.displayHybridBase2();

    hd.displayHybridDerived();

    return 0;

}

OUTPUT:



QUES-6 e) Hierarchal

INPUT: #include <iostream>

using namespace std;

class HierarchicalBase {

public:

    void displayHierarchicalBase() {

        cout << "Hierarchical base class method." << endl;

    }

};

class HierarchicalDerived1 : public HierarchicalBase {

public:

    void displayHierarchicalDerived1() {

        cout << "First hierarchical derived class method." << endl;

    }

};

class HierarchicalDerived2 : public HierarchicalBase {

public:

    void displayHierarchicalDerived2() {

        cout << "Second hierarchical derived class method." << endl;

    }

};

int main() {

    HierarchicalDerived1 h1;

    h1.displayHierarchicalBase();

    h1.displayHierarchicalDerived1();

    HierarchicalDerived2 h2;

    h2.displayHierarchicalBase();

    h2.displayHierarchicalDerived2();

    return 0;

}

OUTPUT:

