Note: Just read all the comments and the explanations, they are for your understanding. You don't need to write them in the journal.

Program 1:

```
#include<iostream>
using namespace std;
class NUMBER {
private:
    short unsigned int num;

    // Private member function to check if the number is palindrome
    int isPalindrome() {
        short unsigned int originalNum = num;
        short unsigned int reversedNum = 0;

        // Reverse the number
        while (originalNum != 0) {
            reversedNum = reversedNum * 10 + originalNum % 10;
            originalNum /= 10;
        }

        // Check if the reversed number is equal to the original number
        return (num == reversedNum) ? 1 : 0;
    }

public:
    // Public member function to display whether the number is palindrome or not
    void display() {
        // Call the private member function 'input' to accept the number
        cout << "Enter a number: ";
        cin >> num;

        // Check if the number is palindrome and display the result
        if (isPalindrome()) {
            cout << num << " is a palindrome." << endl;
        } else {
            cout << num << " is not a palindrome." << endl;
        }
    }
};

int main() {
    // Create an object of the NUMBER class
    NUMBER numObj;

    // Call the public member function 'display' to check if the number is palindrome
    numObj.display();

    return 0;
```

```
}
```

Explanation: This program defines a class NUMBER with a private member num of type short unsigned int. It has a private member function isPalindrome to check whether the number is a palindrome or not. The public member function display calls the input function and then checks and displays whether the entered number is a palindrome or not. The main function creates an object of the NUMBER class and calls the display function to demonstrate the functionality.

Program 2:

```
#include<iostream>
using namespace std;
class FIBO {
private:
   unsigned short int N;

   // Private member function to accept value 'N' from the user
   void input() {
      cout << "Enter the value of N: ";
      cin >> N;
   }

public:
   // Public member function to display the Fibonacci series up to 'N' terms
   void display() {
      // Call the private member function 'input' to accept the value of 'N'
      input();

      unsigned short int term1 = 0, term2 = 1, nextTerm;

      cout << "Fibonacci Series up to " << N << " terms: ";

      for (unsigned short int i = 1; i <= N; ++i) {
         cout << term1 << " ";

         nextTerm = term1 + term2;
         term1 = term2;
         term2 = nextTerm;
      }

      cout << endl;
   }
};

int main() {
   // Create an object of the FIBO class
   FIBO fiboObj;

   // Call the public member function 'display' to show the Fibonacci series
   fiboObj.display();
```

```
    return 0;
}
```

Explanation: This program defines a class FIBO with a private member N of type unsigned short int. It has a private member function input to accept the value of N from the user. The public member function display calls the input function and then displays the Fibonacci series up to the specified number of terms (N). The main function creates an object of the FIBO class and calls the display function to demonstrate the functionality.

Program 3:

```
#include<iostream>
using namespace std;
class PRIME {
private:
    unsigned short int num;

    // Private member function to accept a positive number from the user
    void get_no() {
        do {
            cout << "Enter a positive number: ";
            cin >> num;

            if (num <= 0) {
                cout << "Please enter a positive number." << endl;
            }
        } while (num <= 0);
    }

public:
    // Public member function to check and display whether the num is prime or composite
    void process() {
        // Call the private member function 'get_no' to accept a positive number
        get_no();

        bool isPrime = true;

        if (num <= 1) {
            isPrime = false;
        } else {
            for (unsigned short int i = 2; i <= num / 2; ++i) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime) {
```

```
            cout << num << " is a prime number." << endl;
        } else {
            cout << num << " is a composite number." << endl;
        }
    }
};

int main() {
    // Create an object of the PRIME class
    PRIME primeObj;

    // Call the public member function 'process' to check if the number is prime or composite
    primeObj.process();

    return 0;
}
```

Explanation: This program defines a class PRIME with a private member num of type unsigned short int. It has a private member function get_no to accept a positive number from the user. The public member function process calls the get_no function and then checks and displays whether the number is prime or composite. The main function creates an object of the PRIME class and calls the process function to demonstrate the functionality.

Program 4:

```
#include<iostream>
#include<cmath>
using namespace std;
class ARMSTRONG {
private:
    short unsigned int num;

    // Private member function to check if the number is an Armstrong number
    void process() {
        short unsigned int originalNum, remainder, result = 0;

        originalNum = num;

        while (originalNum != 0) {
            remainder = originalNum % 10;
            result += pow(remainder, 3);
            originalNum /= 10;
        }

        if (result == num) {
            cout << num << " is an Armstrong number." << endl;
        } else {
            cout << num << " is not an Armstrong number." << endl;
        }
    }
```

```
public:
    // Public member function to read a three-digit positive integer and call the process function
    void get_no() {
        do {
            cout << "Enter a three-digit positive integer: ";
            cin >> num;

            if (num < 100 || num > 999) {
                cout << "Please enter a three-digit positive integer." << endl;
            }
        } while (num < 100 || num > 999);

        // Call the private member function 'process' to check if the number is an Armstrong number
        process();
    }
};

int main() {
    // Create an object of the ARMSTRONG class
    ARMSTRONG armstrongObj;

    // Call the public member function 'get_no' to read a three-digit positive integer and check if it's an
Armstrong number
    armstrongObj.get_no();

    return 0;
}
```

Explanation: This program defines a class ARMSTRONG with a private member num of type short unsigned int. It has a private member function process to check if the number is an Armstrong number. The public member function get_no reads a three-digit positive integer from the user and then calls the process function to check if the entered number is an Armstrong number or not. The main function creates an object of the ARMSTRONG class and calls the get_no function to demonstrate the functionality.

Program 5:

```
#include<iostream>
#include<cmath>
using namespace std;
class SERIES {
private:
    unsigned short int N;
    float D;

    // Private member function to compute factorial of a number
    unsigned long long int factorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
```

```
        } else {
            return n * factorial(n - 1);
        }
    }

    // Private member function to compute sin(x) using the given series
    void process() {
        // Convert angle from degrees to radians
        float x = D * (3.14159265359 / 180.0);

        float result = 0.0;
        int sign = 1;

        for (unsigned short int i = 1, j = 1; i <= N; ++i, j += 2) {
            result += sign * (pow(x, j) / factorial(j));
            sign *= -1;
        }

        cout << "sin(" << D << " degrees) = " << result << endl;
    }

public:
    // Public member function to read values for the data members D and N and call the process function
    void get_data() {
        cout << "Enter the angle in degrees (D): ";
        cin >> D;

        cout << "Enter the number of terms (N): ";
        cin >> N;

        // Call the private member function 'process' to compute sin(x) using the given series
        process();
    }
};

int main() {
    // Create an object of the SERIES class
    SERIES seriesObj;

    // Call the public member function 'get_data' to read values and compute sin(x)
    seriesObj.get_data();

    return 0;
}
```

Explanation:
This program defines a class SERIES with private members N and D. It has a private member function process to compute sin(x) using the provided series. The public member function get_data reads values for the data members D and N from the user and then calls the process function to compute and display

sin(x). The main function creates an object of the SERIES class and calls the get_data function to demonstrate the functionality.

Program 6:

```
#include<iostream>
using namespace std;
class SMALLER {
private:
   double d1, d2, d3;

   // Private member function to find the smallest of d1, d2, and d3 and display the result
   void process() {
      double smallest = d1;

      if (d2 < smallest) {
         smallest = d2;
      }

      if (d3 < smallest) {
         smallest = d3;
      }

      cout << "The smallest number is: " << smallest << endl;
   }

public:
   // Parameterized constructor to initialize the private data members
   SMALLER(double val1, double val2, double val3) : d1(val1), d2(val2), d3(val3) {}

   // Public member function to call the private member function process
   void findSmallest() {
      process();
   }
};

int main() {
   // Example usage of the SMALLER class with a parameterized constructor
   double value1, value2, value3;

   cout << "Enter three numbers: ";
   cin >> value1 >> value2 >> value3;

   // Create an object of the SMALLER class with the provided values
   SMALLER smallerObj(value1, value2, value3);

   // Call the findSmallest function to determine and display the smallest number
   smallerObj.findSmallest();

   return 0;
```

```
}
```

Explanation:
In this program:

- The class SMALLER has private data members d1, d2, and d3.
- The private member function process() finds the smallest among d1, d2, and d3 and displays the result.
- The parameterized constructor initializes the private data members with the provided values.
- The findSmallest() function is a public member function that calls the private process() function.
- The main() function takes three double values from the user, creates an object of the SMALLER class, and calls the findSmallest() function to determine and display the smallest number.

Program 7:

```
#include<iostream>
using namespace std;
class BASE {
private:
   float num1;

public:
   float num2;

   // Member function to read data value num1
   void input_data() {
      cout << "Enter the value for num1: ";
      cin >> num1;
   }

   // Member function to return the value of num1
   float get_num1() {
      return num1;
   }
};

class DERIVED : public BASE {
private:
   float sum;

public:
   // Member function to read num2 and compute sum
   void get_data() {
      cout << "Enter the value for num2: ";
      cin >> num2;

      // Call input_data() from the BASE class to read num1
      input_data();

      // Calculate sum by adding num1 and num2
      sum = get_num1() + num2;
```

```
    }

    // Member function to output num1, num2, and sum
    void show_data() {
        cout << "num1: " << get_num1() << endl;
        cout << "num2: " << num2 << endl;
        cout << "sum: " << sum << endl;
    }
};

int main() {
    // Create an object of the DERIVED class
    DERIVED derivedObj;

    // Call the get_data() function to input num2 and compute sum
    derivedObj.get_data();

    // Call the show_data() function to output num1, num2, and sum
    derivedObj.show_data();

    return 0;
}
```

Explanation: C++ program that defines a class BASE with one private data member num1 and one public data member num2. It also defines a public member function input_data() to read the value of num1 and get_num1() to return the value of num1. The program then extends the BASE class to another class DERIVED using public derivation. The DERIVED class has a private data member sum which is calculated by adding num1 and num2. It also has public member functions get_data() to read num2 and compute sum, and show_data() to output num1, num2, and sum. Finally, the program includes a main() function to create an object of type DERIVED and input/output all data.
In this program, DERIVED publicly inherits from BASE. The get_data() function reads num2 and calls input_data() to read num1. It then calculates the sum and the show_data() function outputs all the values. The main() function creates an object of the DERIVED class and calls the necessary functions to demonstrate the functionality.

Program 8:

```
#include<iostream>
using namespace std;
// Abstract class BASE1
class BASE1 {
private:
    int A;

public:
    // Inline parameterized constructor to initialize data member "A"
    inline BASE1(int a) : A(a) {}

    // Inline protected member function that returns the value of "A"
    inline int Get_A() {
```

```
        return A;
    }

    // Pure virtual function (making it an abstract class)
    virtual void display() = 0;
};

// Abstract class BASE2
class BASE2 {
private:
    int B;

public:
    // Inline parameterized constructor to initialize data member "B"
    inline BASE2(int b) : B(b) {}

    // Inline public member function that returns the value of "B"
    inline int Get_B() {
        return B;
    }

    // Pure virtual function (making it an abstract class)
    virtual void display() = 0;
};

// Derived class DERIVED
class DERIVED : public BASE1, protected BASE2 {
private:
    int Z;

public:
    // Inline parameterized constructor to initialize data member "Z"
    inline DERIVED(int a, int b) : BASE1(a), BASE2(b) {
        Z = Get_A() * Get_B();
        show();
    }

private:
    // Inline private member function that displays the value of Z
    inline void show() {
        cout << "Z: " << Z << endl;
    }

public:
    // Public member function to fulfill the pure virtual function in BASE1
    void display() override {
        // Implementation is not needed for this example
    }
};
```

```
int main() {
    // Create an object of the DERIVED class
    DERIVED derivedObj(3, 5);

    return 0;
}
```

Explanation: C++ program that defines the abstract classes BASE1 and BASE2, and a class DERIVED derived from both BASE1 and BASE2. The program includes inline parameterized constructors, inline member functions, and a main() function to demonstrate the functionality.
In this program:

- BASE1 and BASE2 are abstract classes with inline parameterized constructors and inline member functions.
- The DERIVED class is derived from both BASE1 (publicly) and BASE2 (protected), with an inline parameterized constructor to initialize the data member Z.
- The show() function is an inline private member function in DERIVED that displays the value of Z.
- The main() function creates an object of the DERIVED class to demonstrate the functionality.

Program 9:

```
#include<iostream>
using namespace std;
class TIME {
private:
    int Hours;
    int Minutes;

public:
    // Parameterized constructor to initialize data members "Hours" and "Minutes"
    TIME(int h, int m) : Hours(h), Minutes(m) {}

    // Public member function to calculate the summation of two TIME objects
    void Sum(const TIME& t1, const TIME& t2) {
        Hours = t1.Hours + t2.Hours;
        Minutes = t1.Minutes + t2.Minutes;

        // Adjust hours and minutes if necessary
        Hours += Minutes / 60;
        Minutes %= 60;
    }

    // Public member function to display data members of a TIME object
    void Display() {
        cout << "Time: " << Hours << " hours and " << Minutes << " minutes." << endl;
    }
};

int main() {
    // Create two TIME objects with different time values
```

```
    TIME time1(2, 30); // 2 hours and 30 minutes
    TIME time2(1, 45); // 1 hour and 45 minutes

    // Create a third TIME object to store the sum
    TIME sumTime(0, 0);

    // Calculate the sum of time1 and time2 and store it in sumTime
    sumTime.Sum(time1, time2);

    // Display the original time values
    cout << "Time 1: ";
    time1.Display();

    cout << "Time 2: ";
    time2.Display();

    // Display the sum of the two time values
    cout << "Sum of Time 1 and Time 2: ";
    sumTime.Display();

    return 0;
}
```

Explanation: C++ program that defines a class named TIME with the specified members and functionality, including a parameterized constructor, a Sum() member function, and a Display() member function. The main function demonstrates the addition of two TIME objects.

In this program:

- The TIME class has private members Hours and Minutes.
- The parameterized constructor initializes the Hours and Minutes data members.
- The Sum() member function calculates the summation of two TIME objects and assigns the result to the current object.
- The Display() member function displays the Hours and Minutes of a TIME object.
- The main() function creates two TIME objects (time1 and time2), calculates their sum, and displays the original time values along with the sum.

Program 10:

```
#include<iostream>
#include<cmath>
using namespace std;
class VOLUME {
private:
    float r, r1, h, v;  // Data members

public:
    // Parameterized constructor to initialize data member "r"
    VOLUME(float radius) : r(radius), r1(0.0), h(0.0) {
        calculateSphereVolume();
    }
```

```
    // Parameterized constructor to initialize data members "r1" and "h"
    VOLUME(float radius, float height) : r(0.0), r1(radius), h(height) {
        calculateCylinderVolume();
    }

    // Public member function to display volume of sphere and volume of cylinder
    void Display() {
        cout << "Volume of Sphere: " << v << endl;
        cout << "Volume of Cylinder: " << v << endl;
    }

private:
    // Private member function to calculate volume of sphere
    void calculateSphereVolume() {
        v = (4.0 / 3.0) * M_PI * pow(r, 3);
    }

    // Private member function to calculate volume of cylinder
    void calculateCylinderVolume() {
        v = M_PI * pow(r1, 2) * h;
    }
};

int main() {
    // Example usage of the VOLUME class with parameterized constructors
    float radius, height;

    cout << "Enter the radius for the sphere: ";
    cin >> radius;

    // Create an object of VOLUME for a sphere with the provided radius
    VOLUME sphereObj(radius);

    cout << "Enter the radius and height for the cylinder: ";
    cin >> radius >> height;

    // Create an object of VOLUME for a cylinder with the provided radius and height
    VOLUME cylinderObj(radius, height);

    // Display volumes
    cout << "Volumes:" << endl;
    cout << "For Sphere:" << endl;
    sphereObj.Display();

    cout << "For Cylinder:" << endl;
    cylinderObj.Display();

    return 0;
```

}

Explanation: C++ program that defines the class VOLUME with the specified members and functionality, including parameterized constructors and a Display() member function. The main function demonstrates the usage of the class.
In this program:

- The VOLUME class has private data members r, r1, h, and v.
- Two parameterized constructors are defined to initialize the data members for a sphere and a cylinder.
- The Display() member function is a public function to display the volume of the sphere and the volume of the cylinder.
- Two private member functions (calculateSphereVolume() and calculateCylinderVolume()) are defined to calculate the volumes based on the provided parameters.