Q.1. The output of the code will depend on the user input. The **display** function in the **school** class prompts the user to enter the roll number and name, which are then displayed. In the **show** function of the **student** class, the **roll\_no** is set to 1001 and the **name** is set to "Rahul," which is then displayed. So, if the user enters different values for the **roll\_no** and **name** in the **school** class **display** function, those values will be displayed first, followed by the fixed values set in the **student** class **show** function.

Q.2. The output of the program will be:

yamlCopy code

Code : 0 Items : 0 Price : 0.0 Product details - Invent1 type Code : 100 Items : 5 Price : 140.0 Product details - Invent2 type Code : 100 Value : 700

Explanation:

* In the **main** function, an object **s1** of class **invent1** is created with values 100, 5, and 140 for code, items, and price respectively.
* An object **d1** of class **invent2** is created without any arguments, which invokes the default constructor and displays default values (0) for code and value, and then **s1** is assigned to **d1**, which invokes the conversion constructor in **invent2** to convert an object of **invent1** type to **invent2** type.
* The **display** functions in both classes are then called to display the values.

Q.3. The error in the program is in the **friend void show()** function in the **C** class. The **friend** keyword is used to declare a non-member function as a friend of a class. However, in this case, **show** is a member function of the **C** class. To fix the error, remove the **friend** keyword from the **show** function declaration.

Q.4. The output of the program will be:

kotlinCopy code

BaseA constructor called BaseB constructor called Derived's constructor called

Explanation:

* When an object of class **Derived** is created, the constructors of its base classes **BaseA** and **BaseB** are called first, followed by the constructor of **Derived**.

Q.5. The program will not compile due to a typo in the **#include** directive. Instead of **#include<iostram>**, it should be **#include<iostream>**.

Q.6. The output of the program will be **12**, which is the area of the rectangle (**2 \* 2 \* 3 \* 2**). The **set\_values** function is used to set the dimensions of the rectangle, and the **duplicate** function is used to create a new rectangle with double the dimensions of the original rectangle.

Q.7. The error in the program is that there is no overloaded **Add** function that takes two arguments of type **double**. The program calls **Add** with two **int** arguments and then with two **double** arguments, but only the **Add** function with two **int** arguments is defined. To fix the error, define an overloaded **Add** function that takes two **double** arguments.

Q.8. The output of the program will be:

yamlCopy code

5500.26

Explanation:

* The **print** function is overloaded to accept both **int** and **double** arguments. When **print(5)** is called, it prints the integer **5**, and when **print(500.263)** is called, it prints the double **500.263**.

Q.9. The output of the program will be:

Copy code

5 6 7

Explanation:

* The **++** operator is overloaded for the **Overloading** class, which increments the **num** member variable. Initially, **num** is set to **5**. After each **++obj** operation, **num** is incremented by **1**.

Q.10. The output of the program will be:

csharpCopy code

D1 is less than D2

Explanation:

* The **operator<** function is overloaded in the **Distance** class to compare two **Distance** objects based on their **feet** and **inches** values. In this case, **D1** has **feet** = 11 and **inches** = 10, while **D2** has **feet** = 5 and **inches** = 11. Therefore, **D1** is considered less than **D2**.

Q.11. The output of the program will be:

goCopy code

Input two complex number: 1 2 Input two complex number: 3 4 Entered values are: 1 + 2i 3 + 4i Addition of two complex numbers: 4 + 6i Subtraction of two complex numbers: -2 - 2i

Explanation:

* The program creates two objects of the **Complex** class (**x1** and **y1**) and inputs their values using the **input** function.
* It then adds and subtracts these complex numbers using overloaded **+** and **-** operators and displays the results using the **disp** and **print2** functions.

Q.12. The output of the program will be:

Copy code

0

Explanation:

* The **show** function in the **Child** class is trying to access the **data** member of the **Base2** class, which is not initialized. Since **Base2::data** is a private member, it cannot be accessed directly from outside the class, resulting in a compilation error.

Q.13. The output of the program will depend on the size of an integer on your system. It will likely be **20** or **24** (assuming an **int** is **4** bytes and a pointer is **8** bytes). This is because the **GrandChild** class inherits **Base** twice (once through **Child1** and once through **Child2**), which creates two copies of the **Base** class in **GrandChild**.

Q.14. The output of the program will be:

cssCopy code

Inside B

Explanation:

* The **display** function is not defined in class **A**, so the compiler looks for the nearest base class where the function is defined, which is class **B**. Therefore, **C.display()** calls the **display** function of class **B**.

Q.15. The program will not compile due to an attempt to access private members of the base class (**Base**) from the derived class (**Derived**). The **show** function in the **Derived** class tries to access the private members **data1** and **data2** of the **Base** class, which is not allowed.

Q.16. The output of the program will be:

pythonCopy code

This is Base class This is Base class

Explanation:

* The **describe** function takes an argument of type **Base** by value, so when **describe(d)** is called, it slices the **Child** object **d** to a **Base** object, losing the derived class information. This is known as object slicing. Therefore, both calls to **describe** result in calling the **print** function of the **Base** class.

Q.17. The output of the program will be:

Copy code

10 500.263

Explanation:

* The **print** function is overloaded to accept both **int** and **double** arguments. When **print(5)** is called, it prints the integer **5**, and when **print(500.263)** is called, it prints the double **500.263**.