1. What is a virtual base class in C++?
   1. A base class whose objects cannot be created directly.
   2. A base class that can be inherited multiple times in a class hierarchy.
   3. A base class that contains virtual functions.
   4. **A base class that is declared as "virtual."**
2. What is the purpose of a virtual function in C++?
   1. To allow multiple inheritance in classes.
   2. To improve performance in class hierarchies.
   3. **To allow a derived class to provide its implementation.**
   4. To prevent the creation of objects of the base class.
3. When a class is derived from a virtual base class, how many copies of the virtual base class's members will exist?
   1. **Only one copy, shared among all derived classes.**
   2. One copy for each derived class.
   3. Two copies, one for each base class and one for derived class.
   4. The number of copies depends on the inheritance type (public, private, protected).
4. Which keyword is used to declare a virtual function in C++?
   1. **virtual**
   2. override
   3. vfunc
   4. base
5. What happens if a derived class does not override a virtual function of its base class?
   1. The program will not compile.
   2. **The derived class inherits the base class's function implementation.**
   3. The derived class automatically provides a default implementation.
   4. The program will throw a runtime error.
6. In C++, can a virtual function be static?
   1. Yes
   2. **No**
7. Which type of inheritance is required to implement virtual base classes in C++?
   1. Single inheritance
   2. **Multiple inheritance**
   3. Hierarchical inheritance
   4. Multilevel inheritance
8. What is the purpose of the "override" keyword in C++?
   1. **It ensures that a function is implemented in the derived class.**
   2. It declares a virtual function in the base class.
   3. It provides a mechanism for multiple inheritance.
   4. It specifies that a function should be called at runtime.
9. What happens if you try to create an object of an abstract class in C++?
   1. The object will be created successfully.
   2. **The program will not compile.**
   3. The object will have default values for its member variables.
   4. The object will throw an exception at runtime.
10. How can you access a virtual base class's constructor from a derived class?
    1. You cannot access it directly; it is automatically called during object creation.
    2. By using the base class's constructor with parameters.
    3. By using the derived class's constructor with parameters.
    4. **By explicitly calling the base class's constructor using the scope resolution operator (::).**
11. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void display() {

std::cout << "Base Display ";

}

};

class Derived : public Base {

public:

void display() override {

std::cout << "Derived Display ";

}

};

int main() {

Base\* b = new Derived();

b->display();

return 0;

}

* 1. Base Display
  2. **Derived Display**
  3. Compiler Error
  4. Undefined Behavior

1. Can a class have multiple virtual base classes in C++?
   1. **Yes**
   2. No
2. What is the purpose of a pure virtual function in C++?
   1. It is a function without a body, used for documentation purposes.
   2. **It forces a derived class to provide its implementation.**
   3. It is a function that can be accessed directly without creating objects.
   4. It prevents a class from being inherited by any other class.
3. Which type of function cannot be virtual in C++?
   1. Member functions
   2. **Static functions**
   3. Friend functions
   4. Global functions
4. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void print() const {

std::cout << "Base ";

}

};

class Derived : public Base {

public:

void print() const override {

std::cout << "Derived ";

}

};

void display(const Base& obj) {

obj.print();

}

int main() {

Base b;

Derived d;

display(b);

display(d);

return 0;

}

* 1. Base Base
  2. Derived Base
  3. **Base Derived**
  4. Derived Derived

1. Can a virtual function have a different return type in a derived class from that in the base class?
   1. Yes
   2. **No**
2. What happens when a derived class declares a virtual function as non-virtual?
   1. The program will not compile.
   2. **The function in the base class becomes non-virtual as well.**
   3. The function in the derived class will not be accessible.
   4. The function in the base class will be overridden.
3. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void show() {

std::cout << "Base ";

}

};

class Derived : public Base {

public:

void show() override {

std::cout << "Derived ";

}

};

int main() {

Base\* b = new Derived;

Derived\* d = new Derived;

b->show();

d->show();

return 0;

}

* 1. Base Base
  2. **Derived Derived**
  3. Base Derived
  4. Derived Base

1. Which type of function cannot be overridden in C++?
   1. Non-virtual functions
   2. **Static functions**
   3. Public functions
   4. Member functions
2. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void display() {

std::cout << "Base Display ";

}

};

class Derived : public Base {

public:

void display() {

std::cout << "Derived Display ";

}

};

int main() {

Base\* b = new Derived();

Derived\* d = new Derived();

b->display();

d->display();

return 0;

}

* 1. Base Display Base Display
  2. Base Display Derived Display
  3. Derived Display Base Display
  4. **Derived Display Derived Display**

1. In C++, can a base class access a private member of its virtual base class directly?
   1. Yes
   2. **No**
2. Can a derived class have more than one virtual function with the same name and parameters?
   1. **Yes**
   2. No
3. Can a base class be declared as a virtual class in C++?
   1. Yes
   2. **No**
4. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void show() const {

std::cout << "Base ";

}

};

class Derived : public Base {

public:

void show() const override {

std::cout << "Derived ";

}

};

void display(const Base\* obj) {

obj->show();

}

int main() {

const Base\* b = new Derived;

display(b);

return 0;

}

* 1. Base
  2. **Derived**
  3. Base Derived
  4. Derived Base

1. Can a virtual function be a friend of another class?
   1. Yes
   2. **No**
2. What is the output of the following code?

#include <iostream>

class Base {

public:

virtual void display() const {

std::cout << "Base ";

}

};

class Derived : public Base {

public:

void display() const override {

std::cout << "Derived ";

}

};

int main() {

const Base\* b = new Base();

const Derived\* d = new Derived();

b->display();

d->display();

return 0;

}

* 1. Base Base
  2. Derived Derived
  3. **Base Derived**
  4. Derived Base

1. A virtual function is a member function of which of the following class?
   1. Derived class
   2. Parent class
   3. **base class**
   4. Both A and B
2. A virtual function is redefined in which of the following class?
   1. **Derived class**
   2. Parent class
   3. base class
   4. Both A and B
3. The virtual function is used to tell the compiler to perform \_\_\_\_\_\_\_\_?
   1. static linkage
   2. dynamic linkage
   3. late binding
   4. **Both B and C**
4. Which of the following best describes the virtual function?
   1. Function overriding
   2. write a function in the child class that is already present in the parent class
   3. Run time polymorphism
   4. **All of these**
5. In late binding, function call is resolved during?
   1. **runtime**
   2. Compile time
   3. Infinite time
   4. None of these
6. Late binding can be implemented by which of the following?
   1. Function overloading
   2. Function overriding
   3. Virtual function
   4. **B and C**
7. Early binding can be implemented by which of the following?
   1. Virtual function
   2. Function overriding
   3. **Function overloading**
   4. B and C
8. Which statement is true about virtual functions?
   1. Virtual functions can be static members
   2. Virtual functions cannot be static members
   3. Virtual functions must be members of some class
   4. **B and C**