1. What is the other name for early binding?
   1. Static binding
   2. Dynamic binding
   3. Runtime binding
   4. **Compile-time binding**
2. What is the other name for late binding?
   1. Static binding
   2. **Dynamic binding**
   3. Runtime binding
   4. Compile-time binding
3. Early binding is performed at:
   1. **Compile-time**
   2. Link-time
   3. Runtime
   4. Execution time
4. Late binding is performed at:
   1. Compile-time
   2. Link-time
   3. **Runtime**
   4. Execution time
5. Which keyword is used to declare a virtual function in C++?
   1. abstract
   2. **virtual**
   3. override
   4. base
6. Virtual functions are used to achieve:
   1. Early binding
   2. Late binding
   3. **Both early and late binding**
   4. Compile-time errors
7. In C++, a class with at least one pure virtual function is called:
   1. **Abstract class**
   2. Concrete class
   3. Derived class
   4. Static class
8. Can we instantiate an object of an abstract class?
   1. Yes, using the new keyword
   2. Yes, using the malloc function
   3. **No, abstract classes cannot be instantiated**
   4. No, abstract classes are created using templates
9. Which operator is used to access a member function through a pointer to the base class?
   1. ::
   2. .
   3. **->**
   4. \*
10. Which binding is more efficient in terms of performance?
    1. **Early binding**
    2. Late binding
    3. Both have the same performance
    4. It depends on the compiler
11. Which type of binding does function overloading use in C++?
    1. **Early binding**
    2. Late binding
    3. Virtual binding
    4. Static binding
12. Which keyword is used to create an abstract class in C++?
    1. abstract
    2. virtual
    3. abstract\_class
    4. **None, abstract classes are created implicitly**
13. An abstract class can have:
    1. Data members only
    2. Member functions only
    3. **Both data members and member functions**
    4. Neither data members nor member functions
14. When is the memory for a virtual function's vtable allocated?
    1. **Compile-time**
    2. Link-time
    3. Runtime
    4. Execution time
15. Which keyword is used to override a virtual function in a derived class?
    1. **override**
    2. virtual
    3. extends
    4. over
16. A class with a virtual function must have at least one virtual destructor to avoid:
    1. Compile-time errors
    2. Link-time errors
    3. Runtime errors
    4. **Memory leaks**
17. Can a constructor of an abstract class be pure virtual?
    1. Yes
    2. **No**
    3. Only in C++11 and above
    4. Only in C++17 and above
18. Which binding is resolved at the time of function call?
    1. Early binding
    2. **Late binding**
    3. Both early and late binding
    4. None of the above
19. The size of an object of an abstract class is determined by:
    1. The size of its member functions
    2. **The size of its data members**
    3. The size of its virtual functions
    4. The size of the base class
20. How many pure virtual functions must be implemented in a concrete (non-abstract) derived class?
    1. None, pure virtual functions are not implemented
    2. **All pure virtual functions from the base class**
    3. At least one pure virtual function from the base class
    4. At least two pure virtual functions from the base class
21. Can a class have both pure virtual and regular virtual functions?
    1. **Yes, but the class becomes abstract**
    2. Yes, but the class becomes concrete
    3. No, it's not allowed in C++
    4. Only if the class has a default constructor
22. Which function cannot be a virtual function?
    1. Constructors
    2. Destructors
    3. **Static member functions**
    4. Friend functions
23. Can a virtual function be defined in the base class and overridden in the derived class?
    1. **Yes**
    2. No
    3. Only if the base class is abstract
    4. Only if the derived class is abstract
24. When using pointers or references, which binding occurs automatically for virtual functions?
    1. Early binding
    2. **Late binding**
    3. Both early and late binding
    4. None of the above
25. Which keyword is used to invoke the base class version of an overridden virtual function?
    1. base
    2. super
    3. parent
    4. **Using scope resolution operator (::)**
26. If a class inherits an abstract class and does not provide definitions for all pure virtual functions, it becomes:
    1. Concrete class
    2. **Abstract class**
    3. Static class
    4. Interface class
27. Which type of binding is determined by the type of the object pointed to or referenced, not by the type of the pointer or reference?
    1. Early binding
    2. Late binding
    3. **Virtual binding**
    4. Static binding
28. In C++, how can you achieve early binding for a function that is not virtual?
    1. By using the 'final' keyword
    2. **By using the 'static' keyword**
    3. By using the 'const' keyword
    4. Early binding is not possible for non-virtual functions
29. What is the output of the following code?

#include <iostream>

class Shape {

public:

virtual void draw() const {

std::cout << "Drawing shape" << std::endl;

}

};

class Circle : public Shape {

public:

void draw() const override {

std::cout << "Drawing circle" << std::endl;

}

};

int main() {

Circle c;

const Shape\* s\_ptr = &c;

s\_ptr->draw();

return 0;

}

* 1. Drawing shape
  2. **Drawing circle**
  3. Compiler error
  4. Runtime error

1. What is the output of the following code?

#include <iostream>

class A {

public:

virtual void foo() {

std::cout << "A::foo()" << std::endl;

}

};

class B : public A {

public:

void foo() override {

std::cout << "B::foo()" << std::endl;

}

};

int main() {

A\* a\_ptr = new B;

B\* b\_ptr = dynamic\_cast<B\*>(a\_ptr);

if (b\_ptr) {

b\_ptr->foo();

} else {

std::cout << "Dynamic cast failed" << std::endl;

}

return 0;

}

* 1. A::foo()
  2. **B::foo()**
  3. Compiler error
  4. Dynamic cast failed

1. What is the output of the following code?

#include <iostream>

class Vehicle {

public:

virtual void honk() {

std::cout << "Vehicle honk" << std::endl;

}

};

class Car : public Vehicle {

public:

void honk() override {

std::cout << "Car honk" << std::endl;

}

};

void sound(Vehicle& v) {

v.honk();

}

int main() {

Car c;

sound(c);

return 0;

}

* 1. Vehicle honk
  2. **Car honk**
  3. Compiler error
  4. Runtime error