**Q. 1. What is the disadvantage of using #include<bits/stdc++.h> in C++?**

A) It makes the program run slower due to excessive imports.  
B) It increases compilation time significantly since all standard headers are included.  
C) It is not supported by modern C++ compilers.  
D) It reduces the readability of the code by importing unnecessary libraries.

==================================================================

**Q. 2. What is one of the major risks associated with using macros in C++?**

A) Macros are checked for type safety during runtime.  
B) Macros can introduce side effects if their arguments are evaluated more than once.  
C) Macros respect C++ scoping rules.  
D) Macros provide better debugging support than inline functions.

==================================================================

**Q. 3. Encapsulation**

Which of the following best describes encapsulation in object-oriented programming?

A) The process of hiding internal data and restricting access to it.  
B) The ability of a class to inherit properties from another class.  
C) The ability to define functions and variables inside the class.  
D) The process of allowing a class to be used in different contexts.

==================================================================

**Q. 4. Virtual Destructors**

Why should destructors be virtual in base classes when polymorphism is involved?

A) To ensure that the derived class destructor is called when deleting a derived class object through a base class pointer.  
B) To prevent memory leaks in derived classes.  
C) To allow the base class destructor to be overridden in derived classes.  
D) To enable destructor chaining.

==================================================================

**Q. 5. Dynamic Memory Allocation and Polymorphism**

What happens when you allocate memory for a derived class object using a pointer of the base class type, and the base class has a virtual destructor?

A) It will cause a segmentation fault.  
B) The derived class destructor will be invoked, and then the base class destructor.  
C) Only the base class destructor will be invoked.  
D) It will not compile due to type mismatch.

==================================================================

**Q. 6. Virtual Functions and Performance**

When you use virtual functions in a class, which of the following is a potential downside?

A) It increases memory usage due to the vtable mechanism.  
B) It causes compile-time errors.  
C) It decreases the speed of the program due to dynamic dispatch.  
D) Both A and C.

==================================================================

**Q. 7. Overriding vs Overloading**

Which of the following best describes the difference between method overriding and method overloading?

A) Overriding changes the method signature, whereas overloading changes the return type.  
B) Overloading occurs at compile time, whereas overriding happens at runtime.  
C) Overloading happens at runtime, while overriding is a compile-time feature.  
D) Both overloading and overriding are used to handle method calls dynamically at runtime.

==================================================================

**Q. 8. Const-Correctness**

What happens if you pass a const object to a non-const member function?

A) The code will compile and run normally.  
B) The member function can modify the object, even though it is declared as const.  
C) The code will result in a compilation error.  
D) The compiler will automatically convert the object to non-const.

==================================================================

**Q. 9. Dynamic Binding**

In a class hierarchy with virtual functions, what is dynamic binding?

A) The decision about which method to call is made at compile time.  
B) The method to call is resolved based on the type of the object pointed to, not the type of the pointer.  
C) It allows you to call any function dynamically at runtime.  
D) It uses the function with the closest match in the class hierarchy.

==================================================================

**Q. 10. Access Specifiers**

What happens if no access specifier is defined in a C++ class for a member?

A) It is treated as private by default.  
B) It is treated as protected by default.  
C) It is treated as public by default.  
D) It results in a compilation error.

==================================================================

**Q. 11. Composition vs Inheritance**

When would you prefer composition over inheritance in object-oriented design?

A) When objects need to share a common behavior.  
B) When one class needs to own or manage other objects as part of its state.  
C) When multiple classes share a common base class.  
D) When a class has no state and only behavior.

==================================================================

**Q. 12. Run-time Polymorphism**

Which of the following is true about run-time polymorphism in C++?

A) It is achieved through function overloading.  
B) It requires the use of static type checking.  
C) It requires the use of virtual functions in base classes.  
D) It can be achieved only through operator overloading.

==================================================================

**Q. 13. Virtual Functions and Pure Virtual Functions**

What happens when a class contains a pure virtual function?

A) The class cannot be instantiated.  
B) The class will act as a concrete class.  
C) The class can be instantiated but will not have any concrete methods.  
D) The pure virtual function must be defined in the derived class.

==================================================================

**Q. 14. Dynamic Cast**

Which of the following is correct regarding the dynamic\_cast operator in C++?

A) It is used for converting base class pointers to derived class pointers during compile-time.  
B) It throws an exception if the cast is not possible.  
C) It can only be used with objects of class type.  
D) It is used for converting any pointer or reference type to another, regardless of inheritance.

==================================================================

**Q. 15. Abstract Class**

Which of the following statements is true about abstract classes in C++?

A) An abstract class cannot have any member functions.  
B) An abstract class can only have pure virtual functions.  
C) An abstract class can have both pure virtual and concrete functions.  
D) You can create instances of an abstract class directly.

==================================================================

**Q. 16. Exception Handling**

In C++, which of the following operators we can’t overload?

A) +  
B) -  
C) .  
D) %

==================================================================

**Q. 17. Custom Exception Class**

What is the purpose of creating a custom exception class in C++?

A) To define custom error messages and provide more specific exception handling.  
B) To override the default behavior of the std::exception class.  
C) To avoid the use of try and catch blocks.  
D) To automatically handle all types of exceptions.

==================================================================

**Q. 18. Formatted I/O Operations**

Which of the following is a C++ stream manipulator used to set the width of the output?

A) setw  
B) setprecision  
C) fixed  
D) endl

==================================================================

**Q. 19. File Handling**

Which function is used to open a file in C++?

A) open()  
B) ifstream::open()  
C) file.open()  
D) fstream::open()

==================================================================

**Q. 20. Templates**

Which of the following is the correct way to define a template function in C++?

A) template <class T> void func(T arg)  
B) template <T> void func(T arg)  
C) void template <class T> func(T arg)  
D) void func<T>(T arg)

==================================================================

**Q. 21. STL: Stack**

Which of the following C++ Standard Library containers follows the Last In First Out (LIFO) principle?

A) queue  
B) vector  
C) map  
D) stack

==================================================================

### Q. 22. ****Template Specialization****

What happens when you specialize a template function for a specific type in C++?

A) It allows you to implement different functionality for a specific type, overriding the generic template.  
B) It causes a compile-time error unless the template is explicitly instantiated.  
C) It makes the function template private to the class.  
D) It automatically creates a new class for the specialized template.

==================================================================

**Q. 23. Inheritance and Method Overriding**

You have a base class Shape with a virtual method draw() and a derived class Circle that overrides draw(). You create an object of Circle but store it in a pointer of type Shape\*. What happens when you call draw() using the Shape\* pointer?

A) It will call the draw() method of the Shape class.  
B) It will call the draw() method of the Circle class.  
C) It will not compile due to a method signature mismatch.  
D) It will result in a runtime error.

==================================================================

**Q. 24. Abstract Classes**

Consider the following abstract class in C++:

class Shape {

virtual void draw() = 0;

};

What will happen if you try to instantiate the Shape class directly?

A) It will compile but cannot be instantiated.  
B) It will result in a compilation error.  
C) It will run successfully, but will not call draw().  
D) It will give an error at runtime.

==================================================================

**Q. 25. Constructor Initialization**

Consider the following C++ code:

class MyClass {

int x;

public:

MyClass(int i) : x(i) {}

};

What is the purpose of the constructor initialization list (: x(i))?

A) It initializes the value of x with i.  
B) It calls the default constructor for x.  
C) It initializes x after the constructor body is executed.  
D) It does not do anything, and x gets assigned inside the constructor body.

==================================================================

**Q. 26. Multiple Inheritance**

You have two classes ClassA and ClassB, both of which inherit from a base class Base. If ClassA and ClassB both define a method doSomething(), and you create an object of ClassA, which method will be called when you invoke doSomething()?

A) doSomething() from ClassA.  
B) doSomething() from ClassB.  
C) A compile-time error occurs due to ambiguity.  
D) The method will be inherited from Base.

==================================================================

**Q. 27. Polymorphism with Virtual Methods**

Given a base class Animal with a virtual method speak() and derived class Dog that overrides speak(), what is the behavior when a pointer to Animal that points to a Dog object calls speak()?

A) It will always call the speak() method of Animal.  
B) It will call the speak() method of Dog, due to dynamic binding.  
C) It will result in undefined behavior.  
D) It will compile but not run.

==================================================================

**Q. 28. Static Variables in a Class**

Consider a class Counter with a static member variable count that is incremented in the constructor. What happens when you create multiple instances of Counter?

A) Each instance will have its own count value.  
B) All instances will share the same count value.  
C) count will cause a compilation error because it’s static.  
D) The constructor will not execute.

.

==================================================================

**Q. 29. Copy Constructor**

What will happen if you try to copy an object that contains dynamically allocated memory without providing a custom copy constructor?

A) The program will crash due to a shallow copy.  
B) A deep copy of the object will be made automatically.  
C) It will result in undefined behavior.  
D) A compile-time error will occur.

==================================================================

**Q. 30. Operator Overloading**

What is the correct reason to overload the assignment operator in a class?

A) To allow the object to be compared with other objects.  
B) To allocate dynamic memory for each new instance.  
C) To define custom behavior when an object is assigned another object of the same class.  
D) To allow the class to handle multiple data types.

**Q. 31. Friend Functions**

In C++, friend functions can:

A) Access private and protected members of a class.  
B) Only access public members of a class.  
C) Only access static members of a class.  
D) Not be declared within a class.

==================================================================

**Q. 32. Memory Management and Destructor**

What happens if you fail to explicitly define a destructor in a class that dynamically allocates memory?

A) The compiler will automatically generate a destructor to handle the deallocation.  
B) It will result in a memory leak.  
C) The destructor will be inherited from the base class.  
D) The object will be deleted without proper deallocation.

==================================================================

**Q. 33. Run-time Polymorphism**

Which of the following is true about run-time polymorphism in C++?

A) It can be achieved only through function overloading.  
B) It requires the use of static type checking.  
C) It requires the use of virtual functions in base classes.  
D) It can be achieved only through operator overloading.

==================================================================

**Q. 34. Virtual Functions and Pure Virtual Functions**

What happens when a class contains a pure virtual function?

A) The class cannot be instantiated.  
B) The class will act as a concrete class.  
C) The class can be instantiated but will not have any concrete methods.  
D) The pure virtual function must be defined in the derived class.

**Q. 35. Which of the following is true about the delete operator in C++?**

A) It can only be used to free memory allocated by malloc().

B) It does not call the destructor of the object it deallocates.

C) It is used to free memory allocated by new.

D) It is used to deallocate memory on the stack.

**==========================================================**

**Q. 36. Which operator is used for dynamic memory allocation in C++?**

A) malloc()

B) calloc()

C) new

D) alloc()

**==========================================================**

**Q. 37.** What will be the output of the following code

**#include<stdio.h>**

**int main()**

**{**

**int first = 10;**

**int second = 20;**

**int third = 30;**

**{**

**int third = second - first;**

**printf("%d",third);**

**}**

**printf("%d",third);**

**}**

**A) 10 3**

**B) 30 10**

**C) 10 30**

**D) none of these**

**==========================================================**

**==========================================================**

**Q. 38. What will be the output of the code**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int x = 4;**

**if( x == 6);**

**x = 2;**

**cout<<"x = "<<x;**

**return 0;**

**}**

**A) 4**

**B) 6**

**C) 2**

**D) none of these**

**==========================================================**

**Q. 39. When should you use delete[] instead of delete in C++?**

A) When deallocating a dynamically allocated single object.

B) When deallocating a dynamically allocated array.

C) When deallocating memory allocated using malloc().

D) When deleting an object that is referenced by a pointer in the stack.

**==========================================================**

**Q. 40.** What is the primary disadvantage of using inline functions in C++?

a) Inline functions always execute slower than regular functions.  
b) Inline functions increase the binary size because of repeated code.  
c) Inline functions are difficult to debug.  
d) Inline functions cannot be used with templates.

**==========================================================**

**Q. 41. Multiple Inheritance and Ambiguity**

What is one way to resolve ambiguity in multiple inheritance when two base classes have methods with the same name?

A) Use the scope resolution operator.  
B) Use an abstract class.  
C) Use virtual inheritance.  
D) Rename the method in one of the base classes.