C++ Polymorphism Assessment – Section A (MCQs)

Instructions: Circle the correct option. Each question carries 1 mark.

Section A: Multiple Choice Questions (Polymorphism)

1. What does the word "polymorphism" mean in object-oriented programming?

A. Reusing base class variables in multiple classes
B. Having multiple forms of the same function
C. Creating many classes from one object
D. Copying base class functionality directly

2. What is required in C++ to achieve runtime polymorphism?

A. Function overloading
B. Inheriting constructors
C. A virtual function and base pointer/reference
D. Private member functions

3. What does the `virtual` keyword do in a base class method?

A. Prevents function from being overridden
B. Makes the method only usable in the base class
C. Allows derived classes to override the method at runtime
D. Automatically calls the method in the base class

4. What is the purpose of the `override` keyword in C++?

A. Force a function to be private
B. Prevent a function from being used outside the class
C. Mark a function that hides another

D. Tell the compiler we're intentionally overriding a base virtual method

5. What is the output if a base class pointer points to a derived class object, and the function is not marked `virtual` in the base?

A. Base version is called

B. Derived version is called

C. Both are called

D. It's a compile-time error

6. Which of the following is required to make a class abstract in C++?

A. At least one private member

B. At least one static method

C. At least one pure virtual function

D. Inheriting from two base classes

7. What happens when a derived object is passed by value into a function expecting a base class parameter?

A. Polymorphism works as expected

B. Runtime dispatch chooses the right function

C. Object slicing occurs and base function is called

D. Compile-time error

8. Why must destructors be marked `virtual` in base classes when using polymorphism?

A. To allow object slicing

B. To prevent derived class members from being copied

C. To ensure the derived class destructor is called when deleting via base pointer

D. It's not necessary

9. What's the difference between function overriding and function overloading in C++?

A. Overriding is compile-time, overloading is runtime

B. Overloading needs inheritance, overriding does not

C. Overriding replaces base methods, overloading uses different parameters

D. They're the same in C++

10. Which of the following allows a single loop to call the correct `speak()` function for different animals?

A. Overloading constructors

B. Templates

C. `vector<Animal>`

D. `vector<Animal*>` with `virtual speak()` in base