

## **CS 336 – Assignment 7**

### **Question 1**

If a stack's top function returns a pointer instead of a copy, it is not a true data abstraction. According to the notes, a key requirement for abstract data types is that "the representation of objects of the type is hidden from the program units that use these objects". Returning a pointer exposes the internal representation, allowing users to directly access and modify the stack's hidden data without using the provided interface operations like push or pop. This breaks the information hiding principle and makes the program less reliable.

### **Question 2**

C's approach to encapsulation using header files and independent compilation has several dangers. The notes state that "the linker does not check types between a header and associated implementation", which can lead to type mismatches going undetected until runtime. Additionally, C provides no access control mechanism, meaning any program unit including a header can access all global names, violating information hiding. The notes also mention "the inherent problems with pointers" in C, which combined with poor encapsulation can lead to memory corruption.

### Question 3

The arguments for C++ inlining are primarily performance-based, as eliminating function call overhead can significantly speed up small, frequently-called methods. Against inlining, the main concern is potential code bloat from duplicating method code at each call site, which may negatively impact cache performance. The notes don't explicitly discuss inlining but emphasize C++'s focus on efficiency through features like direct memory manipulation and compile-time binding.

### Question 4

The three ways a client can reference a name from a namespace in C++ are: using fully qualified names (`namespace::name`), using declarations (bring specific names into current scope), and using directives (bring all names from namespace into current scope). The notes mention that "C++ Namespaces can place each library in its own namespace and qualify names used outside with the namespace".

### Question 5

According to the notes, subclasses are not subtypes when they don't maintain the "is-a" relationship with their parent class. This can happen when a subclass overrides methods in incompatible ways that change the expected behavior, when it uses private inheritance that hides the parent's

interface, or when it adds functionality inconsistent with the parent's contract and purpose.

## **Question 6**

C++ uses private, protected, and public clauses to control access, with class members defaulting to private access. Java uses individual access modifiers (private, protected, public) per entity with default package-private access. As the notes explain, "Java has a second scoping mechanism, package scope" while C++ provides friend functions for granting special access. The protected modifier also differs between languages in what other classes can access protected members.

## **Question 7**

The notes explain that inheritance addresses two key problems with abstract data types: reusability and organization. ADTs are difficult to reuse as-is and often require creating new, similar types from scratch. Inheritance allows extending existing types with minor changes. Additionally, ADTs exist as independent, flat entities while inheritance creates hierarchical relationships that better model real-world domains.

## Question 8

The notes explain that multiple inheritance in C++ causes problems because classes inherit both interface and implementation from multiple parents, leading to name collisions and complex object layout. Java and C# interfaces avoid these problems because, as the notes state, "An interface can include only method declarations and named constants" with no implementation. This eliminates conflicts from multiple implementations while still allowing a class to support multiple interfaces.