1. What are the arguments for and against representing Boolean values as single bits in memory?
Answer:
For: Boolean values stored as single bits in memory is very space efficient.
Against: Access time to those values is slower than if they were stored as bytes in memory.

2. Compare the tombstone and lock-and-key methods of avoiding dangling pointers, from the points of view of safety and implementation cost.
Answer:

Tombstone	**Lock-and-Key**
Acts as an intermediate between heap dynamic memory and a pointer. Tombstone indicates that variables are no longer available whenever data is deallocated.	Shows pointers as ordered pairs. These ordered pairs contain keys and address where the key is any integer value. The key of its pointer is modified and it holds a value which is different from the variable whenever data is deallocated.
Costly in both time and space because tombstones are never deallocated.	Requires less time and space
No protection from memory access errors. So, less secure.	It uses objects in heap. So, more secure
Doesn’t require additional CPU time.	Requires additional CPU time.
Takes more memory.	Takes less memory.

3. Explain all of the differences between subtypes and derived types.
Answer:
A subtype is compatible with its base type, so we can mix operands of the subtype with operands of the base type. Whereas a derived type is a completely separate type that has the same characteristics as its base type. We cannot mix operands of a derived type with operands of the base type.
Subtypes of a given type will be compatible with each other. A derived type is a new, full-blown type created from an existing one which is incompatible with its parent. However, it inherits the primitive operations defined for the parent type.

4. What significant justification is there for the -> operator in C/C++.
Answer:
If we have a pointer ptr to an object which has a property q. To assign a value to q of that object, we need to do (*ptr).q = value.
Instead of doing that we can easily do ptr->q = value using -> operator which is called arrow operator.

5. What are all of the differences between the enumeration types of C++ and those of Java?
Answer:

C++	**Java**
Enumerations in C++ are internally implemented as variables.	Enumerations in Java are internally implemented as classes. Each enum constant represents an object of type enum.
Get integer values by default.	Doesn’t get default values.
Can’t use them in arithmetic operations.	Can use in arithmetic operations.
It is not recommended to change values once set.	Values can be changed.
There are no methods associated with enum type.	We can implement methods inside enum.