13. What is the difference between type equivalence and type compatibility?

Type Equivalence: User can define types in a language. There are two ways to define type equivalence: structural equivalence and name equivalence. Name equivalence is based on the declarations whereas structural equivalence is based on the notion of meaning of the declarations.  
  
Type Compatibility: Most languages do not require type equivalence; instead they say that type must be compatible with that of context in which it appears.  
  
Type Equivalence:  
The object can be used if its type and the type expected by the context are equivalent, that is, the data type of the object is same as that expected.  
  
Equivalence is a tighter relationship.  
  
Objects and contexts can be equivalent only when their types are same.  
  
Type Compatibility:  
Type Compatibility determines where an objects of a certain type can be used, that is, it tells what kind of data can be held in the object depending on the compatibility of its type.  
  
Compatibility is a looser relationship equivalence.  
  
Objects and contexts can compatible even when their types are different, that is, if they can be used in the same context even after converting them from one type to the other, still they can be called as type compatible.