**Question 1:**

1. Use a concrete example to explain how generic methods can be overloaded:

public bool IsString<T>(T data) where T : class

{

return false;

}

public bool IsString(String data)

{

return true;

}

In this example, the generic method currently returns false when called. However if implemented as a non-generic method as well, it can thus be overloaded. Once overloaded, it will then return true.

1. Explain following terms:
   1. **ICollection**

An interface that is the base for all non-generic collections. Overall, it defines methods that allow it to manipulate collections, such as CopyTo and Count methods.

* 1. **Array**

A collection that stores a fixed set of items of the same type. An array can also be thought of as a collection of variables. Though the variables/items can be changed, the size and type of the array is fixed and cannot be changed.

* 1. **IList**

An interface that implements both ICollection and IEnumerable. Overall, it’s a collection of values that be accessed by their index.

* 1. **Load factor**

A measure of how full the hash table is allowed to get before its size increases. The higher the measure, the higher the collision chance.

* 1. **Hash-table collision**

Collision is when two keys are present in the same location. Collisions cannot be avoided, but can be reduced with good hash functions.

* 1. **Space/time trade-off in hashing**

When the hashing trades memory space for decreased execution time. This results in the algorithm using a lot more space but is overall faster.

* 1. **Dictionary**

A collection of elements which are represented by a field called “key”. A dictionary has operations that allows it to insert, search, and delete pairs in the dictionary.