1.Describe the problem generics address.

**Generics allow developers to create reusable code, and ensure type safe at compile time. Generics can help improve performance by eliminating the need for boxing and unboxing of value types. Flexibility in storing and processing data, and make code easier to read and understand.**

2.How would you create a list of strings, using the generic List class?

**List<string> stringList = new List<string>();**

**List<string> stringList = new List<string> { "hello", "world", "!" };**

3.How many generic type parameters does the Dictionary class have?

**2**

**Dictionary<string, int> myDictionary = new Dictionary<string, int>();**

**In the above example, string is the type of keys and int is the type of values.**

4. True/False. When a generic class has multiple type parameters, they must all match. **F Pair<string, int> myPair = new Pair<string, int>("hello", 123);**

5.What method is used to add items to a List object?

**.Add(T item)**

**List<string> myList = new List<string>();**

**myList.Add("apple");**

**myList.Add("banana");**

**myList.Add("cherry");**

**.Insert(int index, T item)**

**List<string> myList = new List<string>();**

**myList.Insert(0, "apple");**

**myList.Insert(1, "banana");**

**myList.Insert(2, "cherry");**

**.AddRange(IEnumerable<T> collection)**

**List<string> myList = new List<string>();**

**string[] fruits = { "apple", "banana", "cherry" };**

**myList.AddRange(fruits);**

6.Name two methods that cause items to be removed from a List.

**.Remove(T item)**

**List<string> myList = new List<string>() { "apple", "banana", "cherry" };**

**myList.Remove("banana"); // removes "banana" from the list**

**.RemoveAt(int index)**

**List<string> myList = new List<string>() { "apple", "banana", "cherry" };**

**myList.RemoveAt(1); // removes "banana" from the list**

7.How do you indicate that a class has a generic type parameter?

**By using <> angle brackets after the class name**

**For example:**

**MyClass<int> myIntClass = new MyClass<int>();**

**MyClass<string> myStringClass = new MyClass<string>();**

8.True/False. Generic classes can only have one generic type parameter. **F**

9.True/False. Generic type constraints limit what can be used for the generic type. **T**

10.True/False. Constraints let you use the methods of the thing you are constraining to. **T**