CS 6311 – Programming Languages I – Fall 2006
Quiz #9 (20 pts)
Monday, 11/13/6

NAME:

1. True or False. A C# interface class allows the definition of concrete methods. (2 pts)

   False

2. True or False. In C#, if the static data of a class is initialized in a static constructor, the static constructor is implicitly called by the run-time environment. (2 pts)

   True

3. Explain why if a static method uses data members of the class, that those data members must also be declared as static. (3 pts)

   The static methods of a class are accessed via the class itself and not through an instance of the class. With all non-static data not existing until an instance of the object is created a static method having access to non-static data would be attempting to access data that has is not initialized and does not even exist. Therefore, static methods must be declared as static so the data will actually exist and can be initialized before use.

4. In C#, explain the benefit of using a collection that supports the use of generics, such as a List, compared to use a collection that doesn't support the use of generics, such as an ArrayList.

   The use of a data structure that uses generics provides better type checking as only data of the specified type can be put into the data structure, whereas for a non-generic data structure the user is responsible for knowing and maintaining the type of data stored in the data structure. This additional responsibility on the programmer makes it more error prone and less reliable.

5. Given the following partial definition of a Song class written in C#, where the Name property returns the name of the song as a string, and the Bytes property returns the size of song in bytes as an integer. Answer the questions below:

```
class Song : IComparable<Song>
{
        public int CompareTo(Song anotherSong)    {

                if (this.Bytes == anotherSong.Bytes) {

                        return this.Name.CompareTo(anotherSong.Name);

                }  // end if
                else  {

                        return anotherSong.Bytes – this.Bytes;

                } // end else

        }  // end CompareTo method

}  // end class Song
```

a. Investigate the CompareTo method and describe the resulting order of how a List of Song objects stored in the variable mySongs would be sorted by a call to the following: mySongs.Sort(); **Please look at the code carefully**. (5 pts)

   Two level sort: Sorts first in descending order based on the byte size, if the byte size is equal it sorts in ascending alphabetic order.

b. Given a List of Songs called mySongs. Write the line(s) of code in C# to find and print the number of bytes of the largest song in mySongs. Note: the objects in a C# List can be accessed in the same way as a C# ArrayList.(5 pts)

```
mySongs.Sort();
Console.WriteLine(mySongs[0].Bytes);
```