PROGRAMMING ASSIGNMENT #7 - REVIEW

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;

namespace LWTech.ChipAnderson.AccessLogAnalyzer
{

   class Program
   {

      public static void Main()
      {
         Console.WriteLine("Access Log Analyzer");
         Console.WriteLine("====================\n");

         Dictionary<string, int> ipCounts = new Dictionary<string, int>();
         Dictionary<string, int> pathCounts = new Dictionary<string, int>();
         Dictionary<string, int> statusCounts = new Dictionary<string, int>();

         // Read the Access Log File
         try
         {
            using (StreamReader sr = new StreamReader("..//..//..//access-log.txt"))
               while (!sr.EndOfStream)
               {
                  string line = sr.ReadLine();

                  line = line.Replace(" HTTP/1.0", "");
                  line = line.Replace(" HTTP/1.1", "");
                  line = line.Replace("\"", "");

                  string[] tokens = line.Split(' ');

                  string ip = tokens[0];
```

```
            string url = tokens[6];
            string status = tokens[7];

            string path = url.Split('?')[0];

            if (ipCounts.ContainsKey(ip))
                ipCounts[ip]++;
            else
                ipCounts.Add(ip, 1);

            if (pathCounts.ContainsKey(path))
                pathCounts[path]++;
            else
                pathCounts.Add(path, 1);

            if (statusCounts.ContainsKey(status))
                statusCounts[status]++;
            else
                statusCounts.Add(status, 1);

        }
    } catch (IOException ex)
    {
        Console.WriteLine("An filesystem error occurred.  " + ex.Message);
        Console.WriteLine("Unable to continue.");
        return;
    }

    // Display Summary Stats on the Console

    Console.WriteLine("\nStatus Frequencies:");
    Console.WriteLine("====================\n");
    foreach (KeyValuePair<string, int> pair in statusCounts)
        Console.WriteLine(pair.Value + ":\t" + pair.Key);

    Console.WriteLine("\nIP Frequencies:");
    Console.WriteLine("===============\n");
    var ipPairList = new List<KeyValuePair<string, int>>(ipCounts);
    ipPairList.Sort(ReverseValueComparer);                    // Use comparer function for sorting
    foreach (KeyValuePair<string, int> pair in ipPairList)
    {
        if (pair.Value > 9)
```

```
        Console.WriteLine(pair.Value + ":\t" + pair.Key);
}

Console.WriteLine("\nPath Frequencies:");
Console.WriteLine("====================\n");
var pathPairList = new List<KeyValuePair<string, int>>(pathCounts);
pathPairList.Sort((x, y) => y.Value.CompareTo(x.Value));       // Use lambda for sorting
foreach (KeyValuePair<string, int> pair in pathPairList)
{
   if (pair.Value > 9)
      Console.WriteLine(pair.Value + ":\t" + pair.Key);
}

// Write out Summary Stats in CSV format
try
{
   using (StreamWriter sw = new StreamWriter("access-log.csv"))
   {
      sw.WriteLine("Status Frequencies:,");
      foreach (KeyValuePair<string, int> pair in statusCounts)
         sw.WriteLine(pair.Key + "," + pair.Value);
      sw.WriteLine(",,");
      sw.WriteLine("IP Frequencies:,");
      foreach (KeyValuePair<string, int> pair in ipPairList)
      {
         if (pair.Value > 9)
            sw.WriteLine(pair.Key + "," + pair.Value);
      }
      sw.WriteLine(",,");
      sw.WriteLine("Path Frequencies:,");
      foreach (KeyValuePair<string, int> pair in pathPairList)
      {
         if (pair.Value > 9)
            sw.WriteLine(pair.Key + "," + pair.Value);
      }
   }
}
catch (IOException ex)
{
   Console.WriteLine("IO Error writing file.  " + ex.Message);
   Console.WriteLine("Unable to continue.");
   return;
```

```
            }

        }

        private static int ReverseValueComparer(KeyValuePair<string, int> first, KeyValuePair<string, int> second)
        {
            // For reverse sorting KeyValuePairs
            if (first.Value < second.Value) return +1;
            if (first.Value > second.Value) return -1;
            return 0;
        }

    }
}
```