Домашнє завдання №22.1

Застосовуючи FCL скласти програму (C#), яка за допомогою ***List<T>.Sort*** дозволяє з вхідного тексту(***String***) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки ***List***, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у ***Dictionary***.

Застосувати різні способи для виводу результатів роботи програми.

** **коментар:** це завдання тотожне до завдань №19, №20, №21 та №22.2; таким чином можна порівняти різні засоби програмування; далі наводиться приклад повністю виконаного завдання; для компіляції і запуску можна використати https://repl.it/languages/csharp або https://www.tutorialspoint.com/compile_csharp_online.php .*

Вибір варіанту

Задана літера це перша літера прізвища студента *(записаного латинськими літерами)*

Приклад коду

Наведений зразок коду реалізовує завдання з виконання умови розміщення першими слів, що починаються на літеру 'К'.

Літера для прикладу	K
Декларація константи в класі	`const char FIRST_CH = 'K';`

Лістинг

```
using System;
//using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace ACMHW22_1{
    class ACMHW22_1{
        const char FIRST_CH = 'K';

        const int MAX_BUFFER_SIZE = 8192;

        public void scan(String str, List<int> list){
            if (str == null || list == null){
                return;
```

```
            }
            Regex token_re = new Regex("[a-z]+", RegexOptions.IgnoreCase);
            for (Match match = token_re.Match(str); match.Success; match =
match.NextMatch()){
                list.Add(match.Index);
            }
        }

        public static void copyStream(Stream input, Stream output, int start, int
maxExpectedEnd){
            int maxExpectedBytesRead = maxExpectedEnd - start;
            byte[] buffer = new byte[maxExpectedBytesRead];
        for (int bytesRead; (bytesRead = input.Read(buffer, start,
maxExpectedBytesRead)) != 0/* != -1*/ && bytesRead <= maxExpectedBytesRead;){ //
!
           output.Write(buffer, 0, bytesRead);
        }
        }

        public static String toString(int value){
            return String.Format("{0}", value) + "\n";
        }

        public void printListIndexes(List<int> list) {
            if (list == null) {
                return;
            }

            // method 1(by to string function)
            copyStream(
                new MemoryStream(
                    Encoding
                    .ASCII
                    .GetBytes(
                        String
                        .Concat(
                            list
                            .ConvertAll(
                                new Converter<int, String>(
                                    toString
                                )
                             )
                         )
                     )
                ),
                (new StreamWriter(Console.OpenStandardOutput())).BaseStream,
                0,
                MAX_BUFFER_SIZE);

            // method 2
            //list.ForEach(Console.Out.WriteLine); //
```

```
Array.ForEach(list.ToArray(), Console.Out.WriteLine);

            // method 3
            //Console.Out.WriteLine(String.Join("\n", list.ToArray()));

        }

        public void print(String str, List<int> list) {
            if (list == null) {
                return;
            }

            foreach(int value in list){
                String word = str.Substring(value);
                Match match = (new Regex("[a-z]+",
RegexOptions.IgnoreCase)).Match(word);
                if(match.Success) {
                    Console.WriteLine(match.Value);
                }
            }
        }

        class ClassCompareFunction1 : IComparer<int>{
      private String str;

      public ClassCompareFunction1(String text) {
          this.str = text;
        }

      int strcmp__withoutCase(int str1BaseIndex, int str2BaseIndex) {
        for (int str1Index = str1BaseIndex, str2Index = str2BaseIndex; str1Index
< str.Length && str2Index < str.Length; ++str1Index, ++str2Index) {
          char str1_tolower = Char.ToLower(str[str1Index]);
          char str2_tolower = Char.ToLower(str[str2Index]);

          if (str1_tolower != str2_tolower)
          {
            return str1_tolower < str2_tolower ? -1 : 1;
          }
        }

        return 0;
      }

      int strcmp_K__withoutCase(int str1BaseIndex, int str2BaseIndex) {
        char chr1_toupper = Char.ToUpper(str[str1BaseIndex]);
        char chr2_toupper = Char.ToUpper(str[str2BaseIndex]);
        if (chr1_toupper == FIRST_CH && chr2_toupper != FIRST_CH) {
          return -1;
        }
        else if (chr1_toupper != FIRST_CH && chr2_toupper == FIRST_CH) {
```

```
          return 1;
        }
        else if (chr1_toupper == FIRST_CH && chr2_toupper == FIRST_CH) {
          return strcmp__withoutCase(str1BaseIndex + 1, str2BaseIndex + 1);
        }

        return strcmp__withoutCase(str1BaseIndex, str2BaseIndex);
      }
      public int compareFunction(int arg1, int arg2) {
            //return str.Substring(arg1).CompareTo(str.Substring(arg2)); // with
case sensitive
            return String.Compare(str.Substring(arg1), str.Substring(arg2),
StringComparison.OrdinalIgnoreCase);
      }
      public int compareFunction1(int arg1, int arg2) {
        return strcmp_K__withoutCase(arg1, arg2);
      }
        public int Compare(int arg1, int arg2) {
            return compareFunction1((int)arg1, (int)arg2);
        }
    }

        public void sort(String str, List<int> data){
            //IComparer<int> comparer = new ClassCompareFunction1(str);
            data.Sort(new ClassCompareFunction1(str));
        }

        Dictionary<int, String> getMapList(String str, List<int> list){
            if (str == null || list == null){
                return null;
            }

            Dictionary<int, String> mapList = new Dictionary<int, String>();

            for (int index = 0; index < list.Count; ++index){ // .Count()
                String word = str.Substring(list[index]);
                Match match = (new Regex("[a-z]+",
RegexOptions.IgnoreCase)).Match(word);
                if (match.Success){
                    mapList.Add(index, match.Value);
                }
            }

            return mapList;
        }

        public static String toString(KeyValuePair<int, String> data){
            return "{ " + String.Format("{0}", data.Key) + ", " +
String.Format("{0}", data.Value) + " }\n";
        }
```

```
        public void printMapList(Dictionary<int, String> mapList) {
            if (mapList == null) {
                return;
            }

            // By "to string" function
            mapList.Select(toString).ToList().ForEach(Console.Out.Write);
//Console.Out.Write("");

            //// By "to string" Lambda Expression
            //mapList.Select(data => "{ " + String.Format("{0}", data.Key) + ", "
+ String.Format("{0}", data.Value) + " }\n").ToList().ForEach(Console.Out.Write);
// Console.Out.Write;
        }

        static void Main(string[] args){
    ACMHW22_1 acmhw22_1 = new ACMHW22_1();

    List<int> list = new List<int>();
    String text =
      "Sir, in my heart there was a kind of fighting " +
      "That would not let me sleep. Methought I lay " +
      "Worse than the mutines in the bilboes. Rashly- " +
      "And prais'd be rashness for it-let us know " +
      "Our indiscretion sometimes serves us well ... "
      ; // — Hamlet, Act 5, Scene 2, 4–8

        acmhw22_1.scan(text, list);
        acmhw22_1.sort(text, list);
        Dictionary<int, String> mapList = acmhw22_1.getMapList(text, list);

        Console.WriteLine("Indexes:");
        acmhw22_1.printListIndexes(list);

        Console.WriteLine();
        Console.WriteLine("Values:");
        acmhw22_1.print(text, list);

        Console.WriteLine();
        Console.WriteLine("Values(by map):");
        acmhw22_1.printMapList(mapList);

        Console.WriteLine("Press any key to continue . . . ");
        Console.ReadKey();

        }
    }
}
```