Table of Contents

Sort Character & PSBB

This program is built using C# and utilizes .NET 8. It is recommended to use the .NET 8 framework and the commands `dotnet build` and `dotnet run` to compile and run the project. The main class is `Program.cs`, and the additional classes are in the `Class` folder/package.

Usage

```
dotnet build

dotnet run
```

Sort Character (NDL010)

Task

Find the vowel and consonant in words that inputted by user, after that sort the vowel and consonant based on the first appearance.

Input :

```
- One line of words (S).
```

Output :

```
- Contains vowel and consonant characters that has been sorted according to the
following rules.
- Sort the letters according to the order they came out
- Separate between vowels and consonants.
- Process as lowercase letters (whitespaces are ignored)
```

Input	Output
Input one line of words (S) : Sample Case	Vowel Characters : aaee Consonant Characters : ssmplc
Input one line of words (S) : Next Case	Vowel Characters : eea Consonant Characters : nxtcs

Solution

Approach :

1. **Input Handling**: Read the input string and convert it to lowercase to ensure uniformity.
2. **Vowel Processing**: Extract vowels from the input string and sort them based on their first appearance.
3. **Consonant Processing**: Extract consonants from the input string and sort them based on their first appearance.
4. **Output**: Display the sorted vowels and consonants.

Program.CS:

```
///////////////////////////////////////////////////////////////////////////////
        ////////////////////// TASK 1 : Sort Character
//////////////////////////////

///////////////////////////////////////////////////////////////////////////////

        Console.WriteLine("TASK 1 : Sort Character");
        Console.WriteLine("##################################");

        // Calling methods from ShortChar class
        Console.Write("Input one line of words (S) : ");
        string input = Console.ReadLine();
        string charVowel = ShortChar.procVowel(input.ToLower());
        string charConsonant = ShortChar.procConsonant(input.ToLower());

        Console.WriteLine("Vowel Characters : ");
        Console.WriteLine(charVowel);
        Console.WriteLine("Consonant Characters : ");
        Console.WriteLine(charConsonant);

        Console.WriteLine("\n");
```

ShortChar.cs

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

public class ShortChar
{
    public static string procVowel(string param){

        // To save the vowel
        var vowel = new List<char>();
        // To count the occurence using dict
        var countVow = new Dictionary<char, int>();

        for (int i = 0; i < param.Length; i++) {
            char vow = param[i];

            // Condition to check and find the vowel in input, also to exclude the
whitespace
            if (!char.IsWhiteSpace(vow) && vow == 'a' || vow == 'i' || vow == 'u'
|| vow == 'e' || vow == 'o') {
                vowel.Add(vow);
                if (!countVow.ContainsKey(vow)) {
                    countVow[vow] = i;
                }
            }
        }

        // Sort based on occurence
        for (int i = 1; i < vowel.Count; i++) {
            char vow = vowel[i];
            int j = i - 1;

            // Moving the vowel[j] to the correct position based on the occurence
            while (j >= 0 && countVow[vowel[j]] > countVow[vow]) {
                vowel[j + 1] = vowel[j];
                j -= 1;
            }

            vowel[j + 1] = vow;
        }

        return new string(vowel.ToArray());
    }

    public static string procConsonant(string param){

        // To save the consonant
        var consonant = new List<char>();
        // To count the occurence using dict
        var countCons = new Dictionary<char, int>();

        for (int i = 0; i < param.Length; i++) {
            char cons = param[i];

            // Condition to check and find the consonant in input, also to exclude
the whitespace
            if (!char.IsWhiteSpace(cons) && cons != 'a' && cons != 'i' && cons !=
'u' && cons != 'e' && cons != 'o') {
                consonant.Add(cons);
                if (!countCons.ContainsKey(cons)) {
                    countCons[cons] = i;
                }
            }
        }

        // Sort based on occurence
        for (int i = 1; i < consonant.Count; i++) {
            char cons = consonant[i];
            int j = i - 1;

            // Moving the vowel[j] to the correct position based on the occurence
            while (j >= 0 && countCons[consonant[j]] > countCons[cons]) {
                consonant[j + 1] = consonant[j];
                j -= 1;
            }

            consonant[j + 1] = cons;
        }

        return new string(consonant.ToArray());
    }
}
```

1. **Vowel Processing**
    - Iterating through each character in the input string.
    - It will check if the character is a vowel and not a whitespace.
    - After that adding the vowel to a list and track its first occurrence using a dictionary.
    - And sorting the vowel list based on their first occurrence using the dictionary.
2. **Consonant Processing**
    - Similar to vowel processing, it will iterating through each character.

- Checking if the character is a consonant and not a whitespace.
- Add the consonant to a list and track its first occurrence using a dictionary.
- Sorting the consonant list based on their first occurrence using the dictionary.

PSBB (Pembatasan Sosial Berskala Besar) (NDL011)

Task :

Galih and Ratna married during the COVID 19 period and only invited the families of both partners. they rented a number of minibuses to pick up all of their families to go to the wedding.

But during COVID 19, the government held a PSBB program to reduce the impact of the spread of the virus. Each mini bus can only carry at most 4 passengers.

What a minimum number of buses will they need to rent if all members of each family should ride in the same Busses. (one bus can't take more than two family)

Input :

```
- The first line contains integer n — the number of families.
- The second line contains a sequence of integers. The integers are separated by a
space, each integer is the number of members in the family.
```

Output :

```
- Print the single number — the minimum number of buses necessary to drive all
family to the Wedding.
- Print “Input must be equal with count of family” if input number of family is
not equal with count of family.
```

Input	**Output**
Input the number of families : 5 Input the number of members in the family (separated by a space) : 1 2 4 3 3	Minimum bus required is : 4
Input the number of families : 8 Input the number of members in the family (separated by a space) : 2 3 4 4 2 1 3 1	Minimum bus required is : 5
Input the number of families : 5 Input the number of members in the family (separated by a space) : 1 5	Input must be equal with count of family

Solution

Approach to find the solution :

1. Parse the input of number of families to integer because the input is in String format
2. Split the input of number of members in the family (separated by a space) and store it in a list
3. Sort the families by the number of members.
4. Pair families to fit into as few buses as possible.

Program.cs:

```
using System;

//////////////////////////////////////////////////////////////////////////////
/////////////// TASK 2 : PSBB ( Pembatasan Sosial Berskala Besar ) ///////////
//////////////////////////////////////////////////////////////////////////////
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("TASK 2 : PSBB - Pembatasan Sosial Berskala Besar");
            Console.WriteLine("#################################");

            Console.Write("Input the number of families : ");
            int inputNum = int.Parse(Console.ReadLine());

            Console.Write("Input the number of members in the family (separated by
a space) : ");
            string[] inputFamily = Console.ReadLine().Split();

            // Validating input length
            if (inputFamily.Length != inputNum) {
                Console.WriteLine("Input must be equal with count of family");
                return;
            }

            string result = PSBB.MiniBus(inputNum, inputFamily);
            Console.WriteLine(result);
        }
    }
```

PSBB.cs:

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

public class PSBB
{
    public static string MiniBus(int count, string[] membersTotal){

        // Parsing the string in List membersTotal to int
        int[] members = new int[count];
        for (int i = 0; i < count; i++) {
            members[i] = int.Parse(membersTotal[i]);
        }

        // Sorting the family member
        Array.Sort(members);

        int busNeeded = 0;
        int x = 0;
        int y = members.Length - 1;

        // Group the family
        while (x <= y) {

            // This check if both small & large family can be in same bus
            if (members[x] + members[y] <= 4) {
                x += 1;
            }
            y -= 1;

            busNeeded += 1;
        }

        return $"Minimum bus required is : {busNeeded}";
    }
}
```

1. **Sorting**:
    - The purpose of this sorting is to facilitate bigger and smaller family to be paired together.
2. **Pairing**:
    - Use two pointers, one starting from the beginning is var x(smallest family) and one from the end is y(largest family).
    - Pair the smallest and largest families together to maximize bus usage.
    - If the smallest and largest families together fit into one bus (4 members or fewer), move both pointers inward.
    - If not, just move the pointer of the largest family.

Using this the families will be pair and usage the bus become efficient also can find the bus that needed to use.