Name: Kevin Octavianus Halim

My brief explanation of my codes for Case 1 and Case 2 are on the next pages after I show the results for every cases.

The Main class

```
public static void MainMenu()
{
    bool status = true;
    while (status)
    {
        try
        {
            Console.WriteLine("Select Case Number ( 1 - 2 ), Type 3 For Exit The Program");
            int caseNo = Convert.ToInt32(Console.ReadLine());
            switch (caseNo)
            {
                case 1:
                    Case1();
                    break;
                case 2:
                    Case2();
                    break;
                case 3:
                    Console.WriteLine("\nExit The Program...");
                    status = false;
                    break;

                default:
                    Console.WriteLine("\n### Invalid Input Please Input Number Between 1 - 3 ###\n");
                    break;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine($"{e.Message.ToString()}\n");
        }
    }
}

public static void Main()
{
    MainMenu();
}
```

I made the Menu System, so the users able to navigate through cases without having to restarted the program.

The Output of the Main Class

```
Microsoft Visual Studio Debug Console
Select Case Number ( 1 - 2 ), Type 3 For Exit The Program

Input string was not in a correct format.

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
5

### Invalid Input Please Input Number Between 1 - 3 ###

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
3

Exit The Program...

D:\Code\NET\ConsoleApp\bin\Debug\net6.0\ConsoleApp.exe (process 16800) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

I add some logics to make sure the input is valid, like if the input is null or empty, or not between 1 – 3 the program will keep asking the user for inputting a valid number.

Case No.1 Result

```
D:\Code\NET\ConsoleApp\bin\Debug\net6.0\ConsoleApp.exe
Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
1

--- This is Case Number 1 - Sort Character ---

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

 Program Done

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
1

--- This is Case Number 1 - Sort Character ---

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

 Program Done

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
1

--- This is Case Number 1 - Sort Character ---

Input one line of words (S) : AIUEO auieo
Vowel Characters :
aaiiuueeoo
Consonant Characters :


 Program Done

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
1

--- This is Case Number 1 - Sort Character ---

Input one line of words (S) : Case Testing
Vowel Characters :
aeei
Consonant Characters :
cssttng

 Program Done
```

I tried testing with another words to check if my code is working properly as expected and not just for specific words.

Case No. 2 Result

```
Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
2

--- This is Case Number 2 - PSBB ( Pembatasan Sosial Berskala Besar ) ---

Input the number of families : 5
--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 1 2 4 3 3
Minimum bus required is : 4


 Program Done

Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
2

--- This is Case Number 2 - PSBB ( Pembatasan Sosial Berskala Besar ) ---

Input the number of families : 8
--- Total 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


 Program Done
```

If the family member doesn't equals to count of family (total family)

```
Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
2

--- This is Case Number 2 - PSBB ( Pembatasan Sosial Berskala Besar ) ---

Input the number of families : 5
--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 1 5

### Input must be equal with count of family ###

--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 1

### Input must be equal with count of family ###

--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 0

### Input must be equal with count of family ###

--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 1 5 7 4 6 5 4

### Input must be equal with count of family ###

--- Total Families = 5 ---
Input the number of members in the family
(separated by a space) : 1 5 2 2 3
Minimum bus required is : 4


 Program Done
```

The Program will not stop and instead it will ask the user to re-input the total family until user input it correctly.

Case No.1 Code Overview

```
public static char[] vowels = { 'a', 'i', 'u', 'e', 'o' };

    public static string procVowel(string param)
    {
        //Initialize StringBuilder for better string manipulation
        StringBuilder vowel = new StringBuilder();

        //Remove any spaces and change the param/input to LowerCase
        param = param.Replace(" ", "").ToLower();

        //looping for finding and append/insert vowels inside the input
        foreach (char p in param)
        {
            if (vowels.Contains(p))
            {
                vowel.Append(p);
            }
        }

        //Sorting the vowels by its order of appearance in the input
        string sortedVowels = new string(vowel
            .ToString()
            .ToCharArray()
            .OrderBy(p => param.IndexOf(p))
            .ToArray());

        return sortedVowels;
    }

    public static string procConsonant(string param)
    {

        StringBuilder consonant = new StringBuilder();
        param = param.Replace(" ", "").ToLower();

        //Same logic & result as in the procVowel but use LINQ
        foreach (var p in
            from char p in param
            where !vowels.Contains(p)
            select p)
        {
            consonant.Append(p);
        }


//Sorting the consonants by its order of appearance in the input
        string sortedConsonants = new string(consonant
            .ToString()
            .ToCharArray()
            .OrderBy(p => param.IndexOf(p))
            .ToArray());

        return sortedConsonants;
    }
```

I decided to use LINQ in the procConsonant() to showcase that I'm familiar with LINQ in C#

```
public static void Case1()
{
    try
    {
        Console.WriteLine("\n--- This is Case Number 1 - Sort Character ---\n");
        Console.Write("Input one line of words (S) : ");
        string input = Console.ReadLine();

        string charVowel = procVowel(input);
        string charConsonant = procConsonant(input);

        Console.WriteLine("Vowel Characters : ");
        Console.WriteLine(charVowel);
        Console.WriteLine("Consonant Characters : ");
        Console.WriteLine(charConsonant);
        Console.WriteLine("\n Program Done \n");
    }
    catch (Exception e)
    {
        Console.WriteLine($"{e.Message.ToString()}\n");
    }
}
```

I add Try Catch for handling the user input, just in case if there are something or any input that I can't predicted and try to make the program not stopping when encountering errors.

Case No. 2 Code Overview

```
public static void Case2()
{
    Console.WriteLine("\n--- This is Case Number 2 - PSBB ( Pembatasan Sosial Berskala Besar ) ---\n");

    int busCapacity = 4;
    bool validInput = false;
    int totalPassengers = 0;
    int totalFamily = 0;
```

Initialization of some variables especially for validInput boolean and totalFamily int for conditional looping.

```
    try
    {
        while (totalFamily <= 0)
        {
            Console.Write("Input the number of families : ");
            totalFamily = Convert.ToInt32(Console.ReadLine());
            if (totalFamily <= 0)
            {
                Console.WriteLine("\n### Total Family Must Be At Least 1 ###\n");
            }
        }
```

The Next Step is started with Try Catch, and inside the Try block code, I use looping to make sure user didn't input Total Family Below or Equals to 0 (negative number and literal 0).

```
            while (!validInput)
            {
                {
                    Console.Write($"--- Total Families = {totalFamily} ---\n");
                    Console.Write("Input the number of members in the family\r\n(separated by a space) : ");
                    string[] members = Console.ReadLine().Split(' ');

                    //Handler to find if user input member = 0
                    if (members.Any(m => m.Contains("0")))
                    {
                        Console.WriteLine("\n### Member must not be empty ###\n");
                    }
                    //Handler to make sure the input count must be equal or same as total family
                    else if (members.Length != totalFamily)
                    {
                        Console.WriteLine("\n### Input must be equal with count of family ###\n");
                    }
                    else
                    {
                        //loop for converting string to int for counting totalPassenger
                        foreach (string m in members)
                        {
                            totalPassengers += int.Parse(m);
                        }
                        validInput = true;
                    }
                }
            }
            /*
            Rounding up for totalBus required for picking up passengers,
            since there might be a bus where it consist less than 4 people inside
             */
            int totalBus = (int)Math.Ceiling((double)totalPassengers / busCapacity);

            Console.WriteLine($"Minimum bus required is : {totalBus} \n");

            Console.WriteLine("\n Program Done \n");
        }
        catch (Exception e)
        {
            Console.WriteLine($"{e.Message.ToString()}\n");
        }
    }
```

The program will keep looping as long as User Inputs are invalid, such as if user input 0 when asked to input number of members in the family,
and also if the total input doesn't equals to count of totalFamily from previous code.

The Result of the error handler I made will be:

```
Select Case Number ( 1 - 2 ), Type 3 For Exit The Program
2

--- This is Case Number 2 - PSBB ( Pembatasan Sosial Berskala Besar ) ---

Input the number of families : 2
--- Total Families = 2 ---
Input the number of members in the family
(separated by a space) : 0 0

### Member must not be empty ###
```