INSPERITY

CODE CHALLENGE

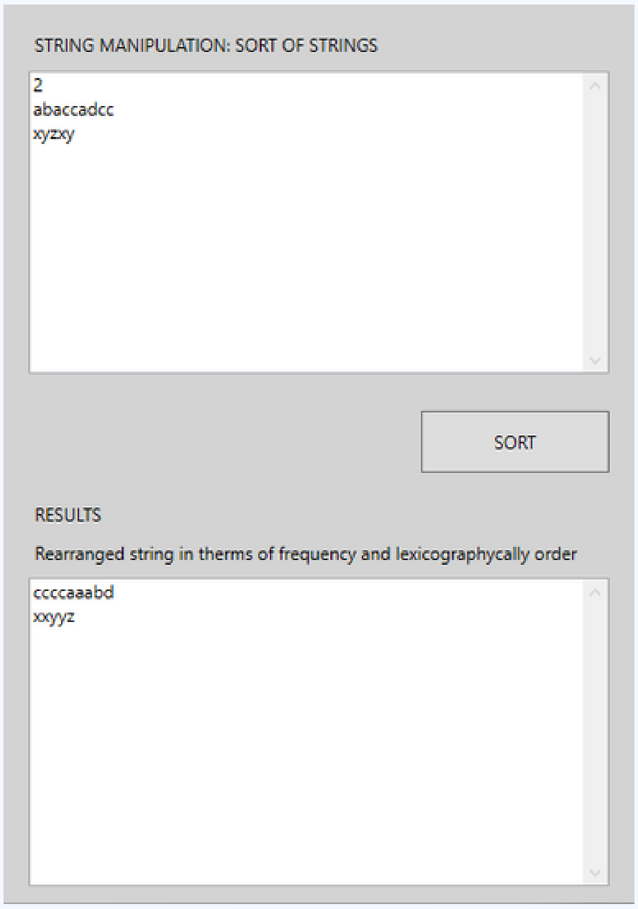
STRING MANIPULATION: SORT OF STRINGS

POSTULANT:

JUAN JOSE MAYORA

SCREEN CAPTURE OF TEST CASES

2



STRING MANIPULATION: SORT OF STRINGS

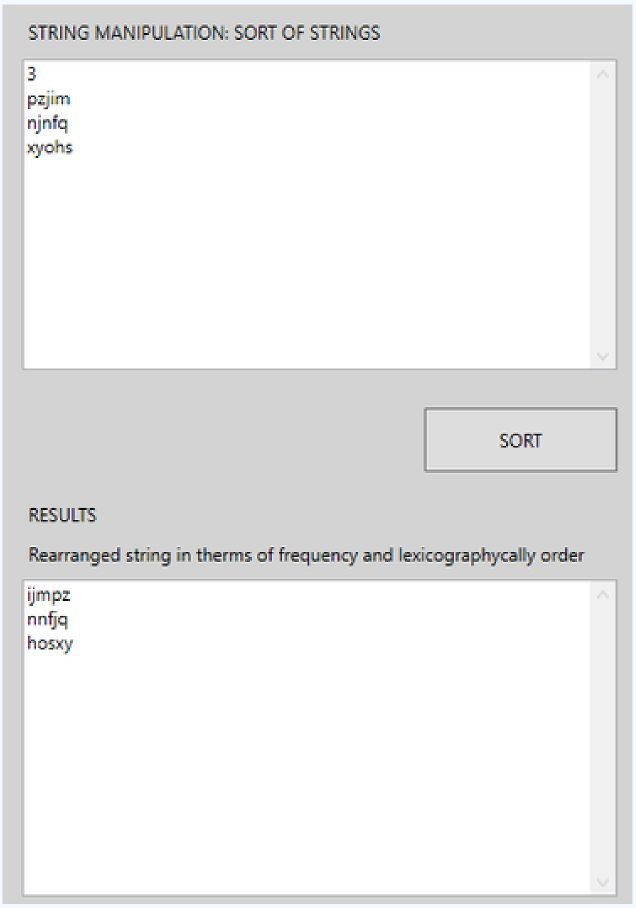
```
2
abaccadcc
xyzxy
```

RESULTS

Rearranged string in therms of frequency and lexicographycally order

```
ccccaaabd
xxyyz
```

3



STRING MANIPULATION: SORT OF STRINGS

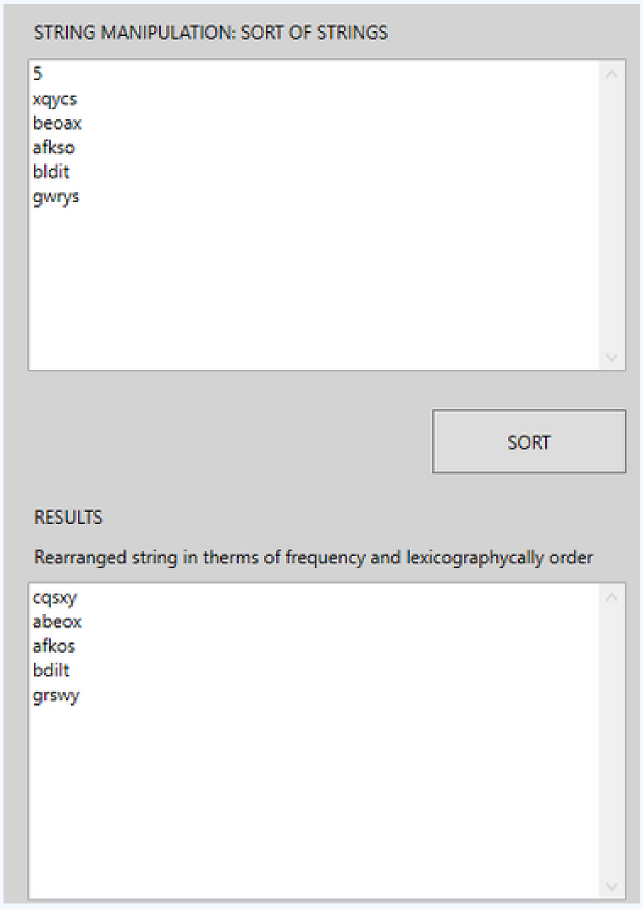
```
3
pzjim
njnfq
xyohs
```

RESULTS

Rearranged string in therms of frequency and lexicographycally order

```
ijmpz
nnfjq
hosxy
```

5



STRING MANIPULATION: SORT OF STRINGS

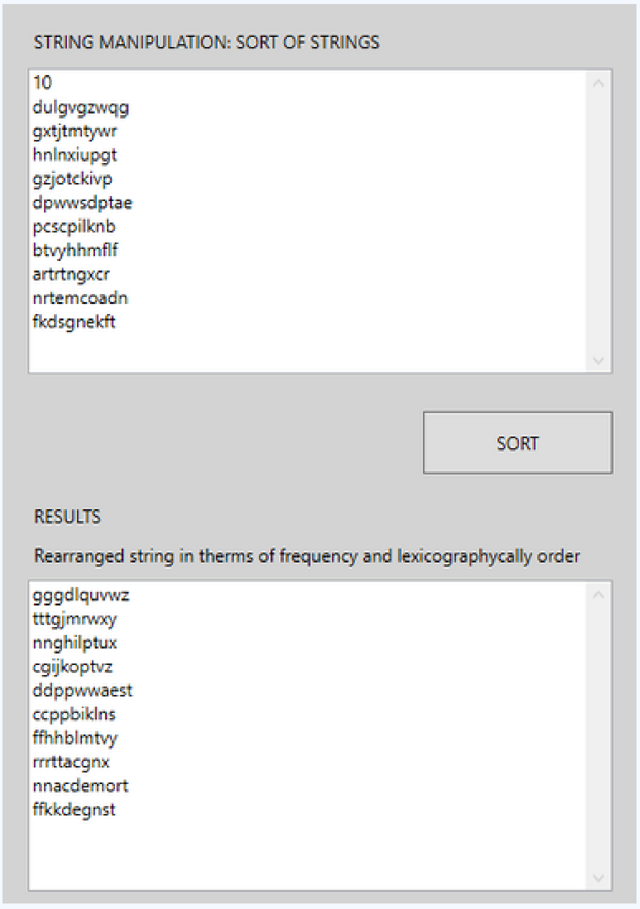
```
5
xqycs
beoax
afkso
bldit
gwrys
```

RESULTS

Rearranged string in therms of frequency and lexicographycally order

```
cqsxy
abeox
afkos
bdilt
grswy
```

10



STRING MANIPULATION: SORT OF STRINGS

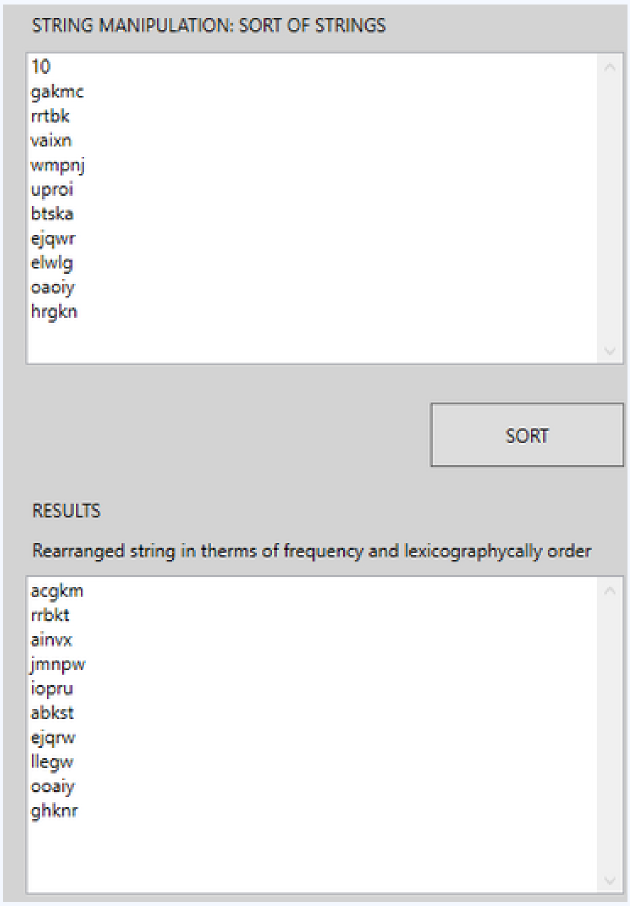
```
10
dulgvgzwqg
gxtjtmtywr
hnlnxiupgt
gzjotckivp
dpwwsdptae
pcscpilknb
btvyhhmflf
artrtngxcr
nrtemcoadn
fkdsgnekft
```

RESULTS

Rearranged string in therms of frequency and lexicographycally order

```
gggdlquvwz
tttgjmrwxy
nnghilptux
cgijkoptvz
ddppwwaest
ccppbiklns
ffhhblmtvy
rrrttacgnx
nnacdemort
ffkkdegnst
```

10



STRING MANIPULATION: SORT OF STRINGS

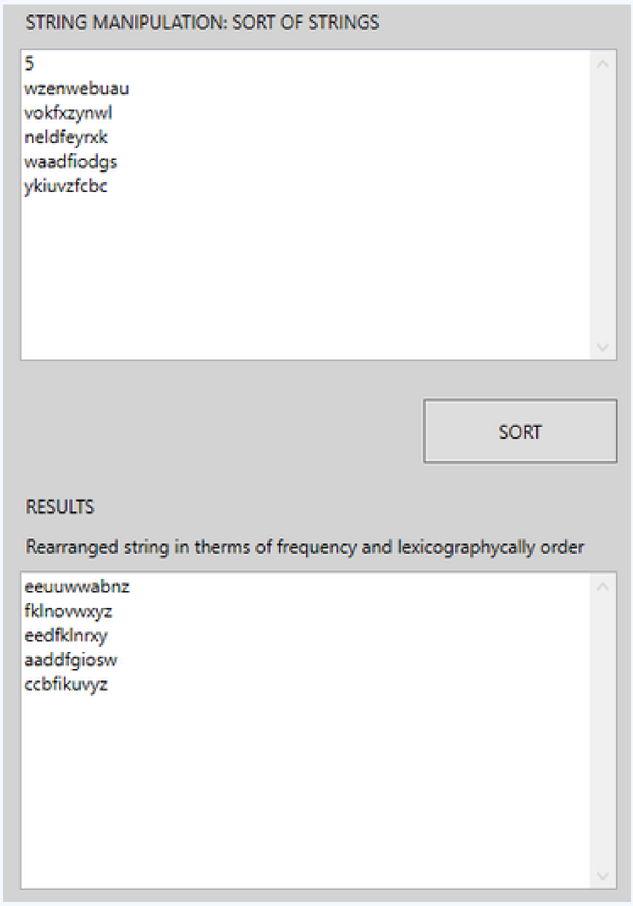
```
10
gakmc
rrtbk
vaixn
wmpnj
uproi
btska
ejqwr
elwlg
oaoiy
hrgkn
```

RESULTS

Rearranged string in therms of frequency and lexicographycally order

```
acgkm
rrbkt
ainvx
jmnpw
iopru
abkst
ejqrw
llegw
ooaiy
ghknr
```

5



STRING MANIPULATION: SORT OF STRINGS

```
5
wzenwebuau
vokfxzynwl
neldfeyrxk
waadfiodgs
ykiuvzfcbc
```

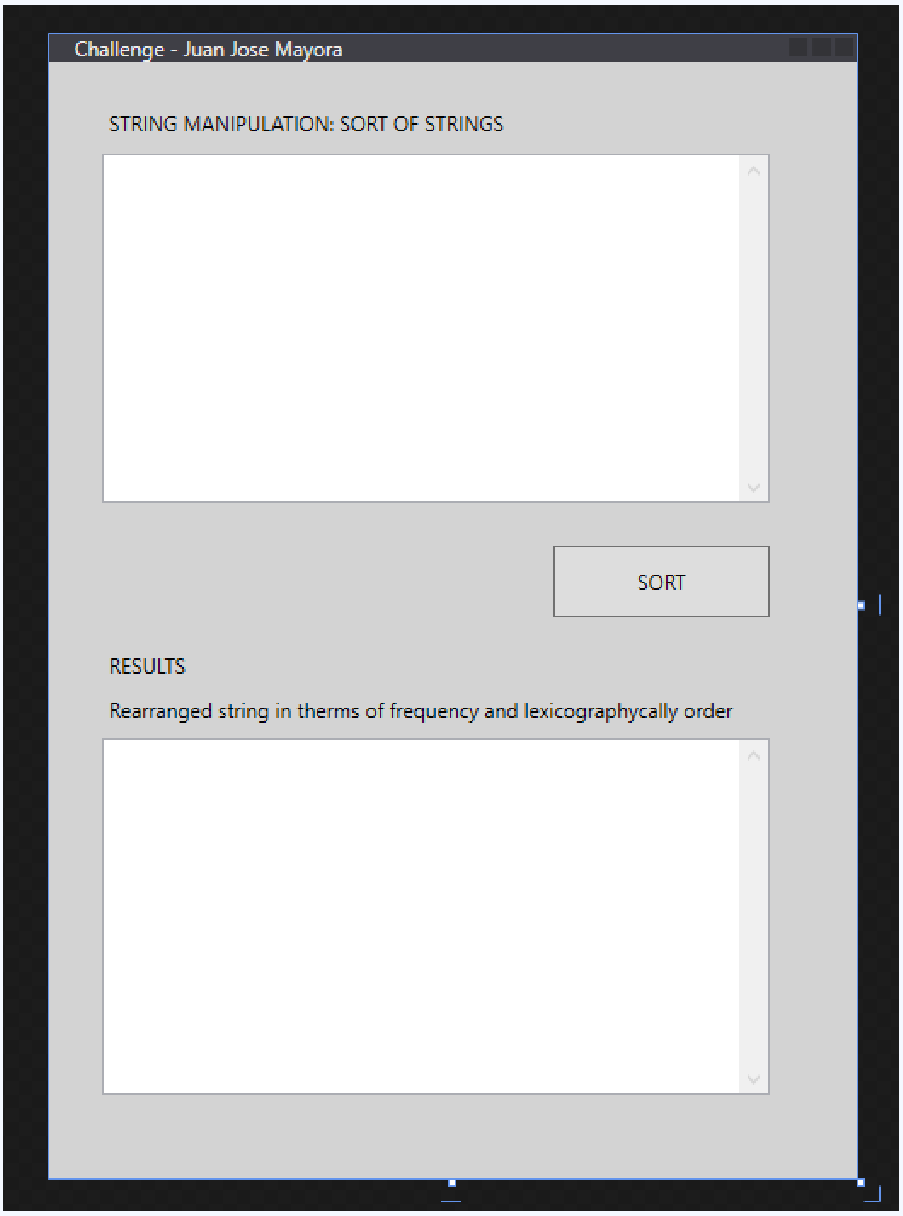
RESULTS

Rearranged string in therms of frequency and lexicographycally order

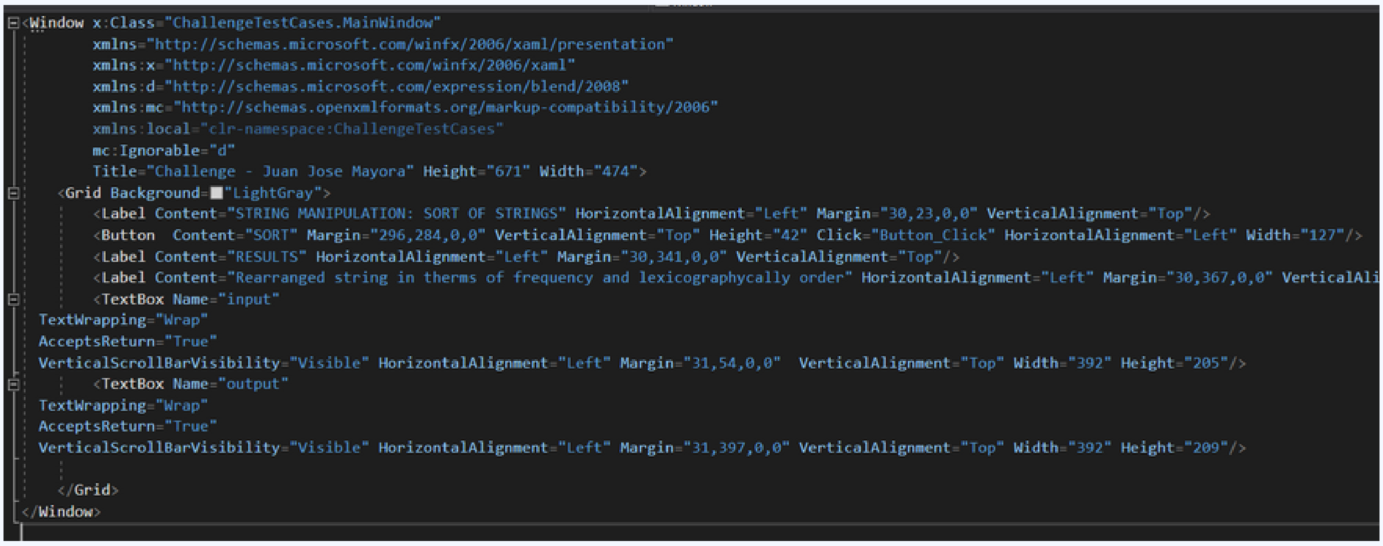
```
eeuuwwabnz
fklnovwxyz
eedfklnrxy
aaddfgiosw
ccbfikuvyz
```

CODE NOTES

USER INTERFACE


Challenge - Juan Jose Mayora
STRING MANIPULATION: SORT OF STRINGS
SORT
RESULTS
Rearranged string in therms of frequency and lexicographycally order

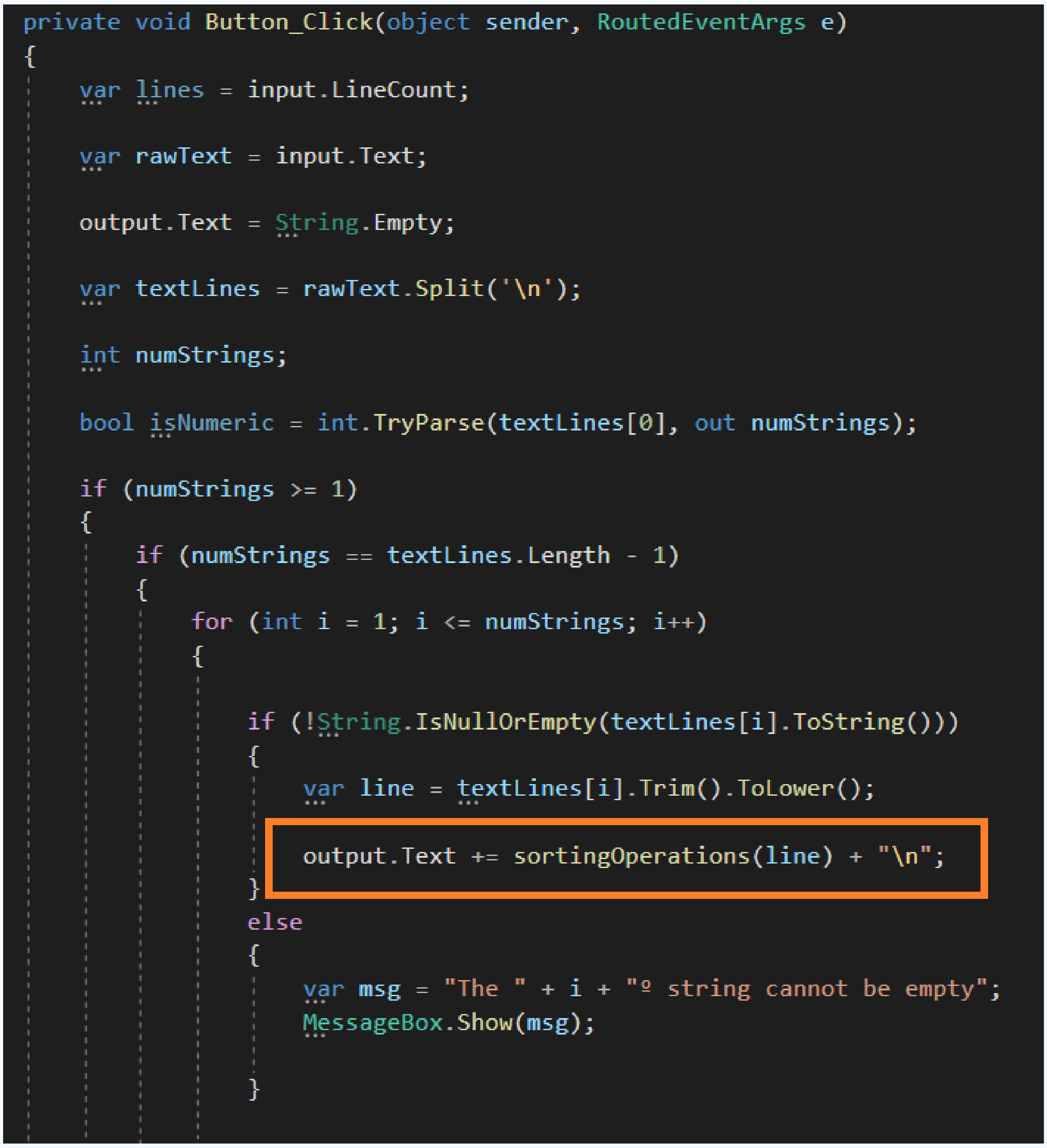
XAML CODE



```xml
<Window x:Class="ChallengeTestCases.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:ChallengeTestCases"
        mc:Ignorable="d"
        Title="Challenge - Juan Jose Mayora" Height="671" Width="474">
    <Grid Background="LightGray">
        <Label Content="STRING MANIPULATION: SORT OF STRINGS" HorizontalAlignment="Left" Margin="30,23,0,0" VerticalAlignment="Top"/>
        <Button  Content="SORT" Margin="296,284,0,0" VerticalAlignment="Top" Height="42" Click="Button_Click" HorizontalAlignment="Left" Width="127"/>
        <Label Content="RESULTS" HorizontalAlignment="Left" Margin="30,341,0,0" VerticalAlignment="Top"/>
        <Label Content="Rearranged string in therms of frequency and lexicographycally order" HorizontalAlignment="Left" Margin="30,367,0,0" VerticalAli
        <TextBox Name="input"
  TextWrapping="Wrap"
  AcceptsReturn="True"
  VerticalScrollBarVisibility="Visible" HorizontalAlignment="Left" Margin="31,54,0,0"  VerticalAlignment="Top" Width="392" Height="205"/>
        <TextBox Name="output"
  TextWrapping="Wrap"
  AcceptsReturn="True"
  VerticalScrollBarVisibility="Visible" HorizontalAlignment="Left" Margin="31,397,0,0" VerticalAlignment="Top" Width="392" Height="209"/>

    </Grid>
</Window>
```

CODE NOTES

BUTTON CLICK EVENT CALLS THE FUNCTION

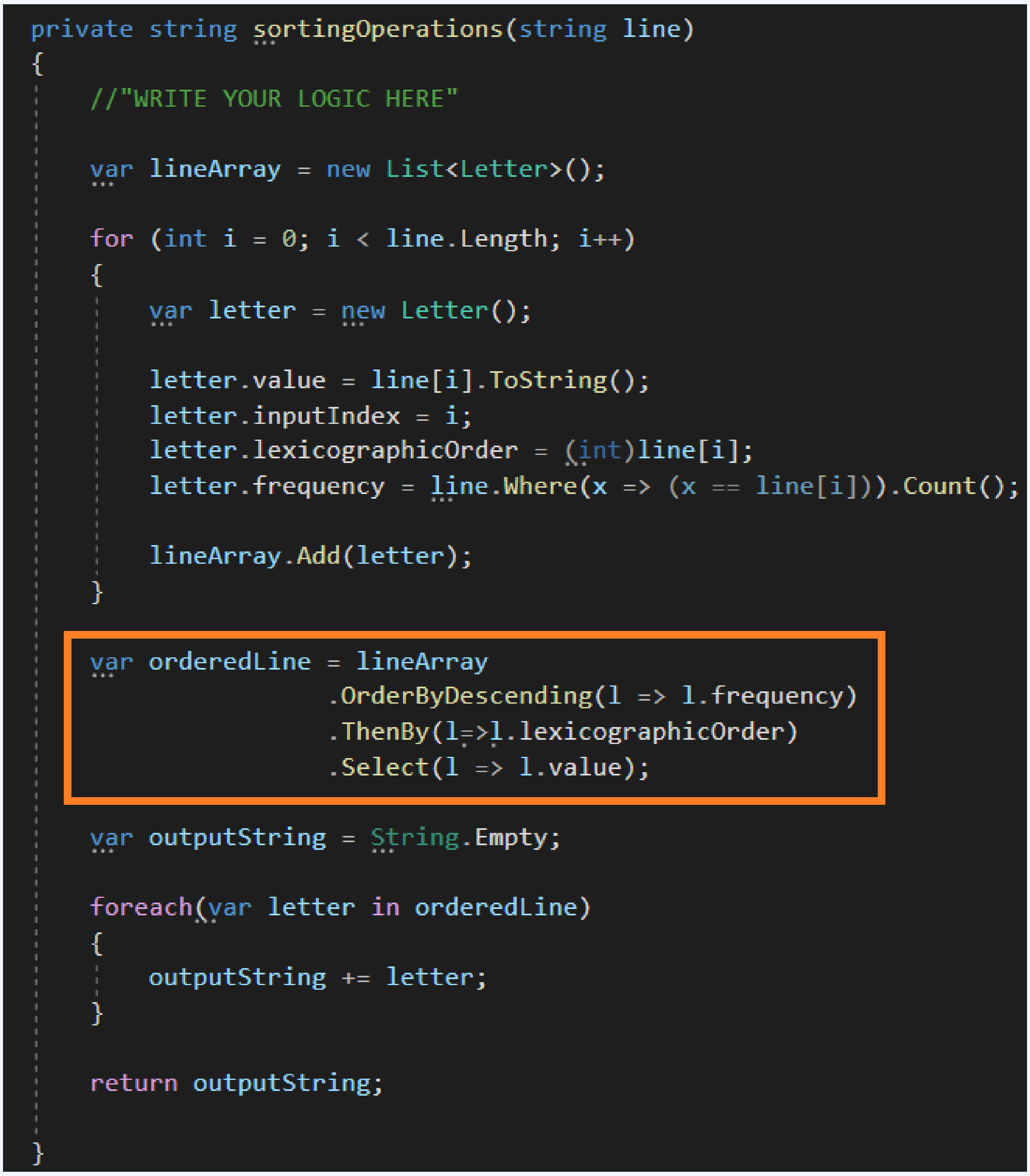


```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var lines = input.LineCount;

    var rawText = input.Text;

    output.Text = String.Empty;

    var textLines = rawText.Split('\n');

    int numStrings;

    bool isNumeric = int.TryParse(textLines[0], out numStrings);

    if (numStrings >= 1)
    {
        if (numStrings == textLines.Length - 1)
        {
            for (int i = 1; i <= numStrings; i++)
            {

                if (!String.IsNullOrEmpty(textLines[i].ToString()))
                {
                    var line = textLines[i].Trim().ToLower();

                    output.Text += sortingOperations(line) + "\n";
                }
                else
                {
                    var msg = "The " + i + "º string cannot be empty";
                    MessageBox.Show(msg);

                }
```

AFTER SOME VALIDATIONS, THE BUTTON CLICK EVENT CALLS THE FUNCTION AND SET ITS RETURN VALUE TO THE OUTPUT TEXTBOX

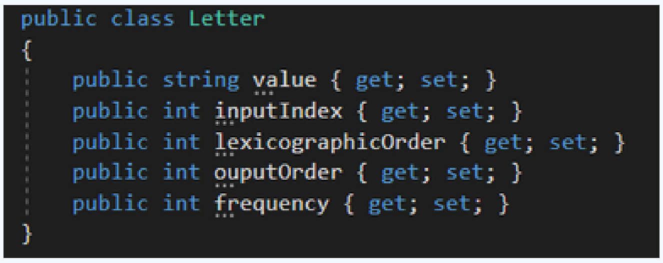
CODE NOTES

THE FUNCTION: SORTING OPERATIONS



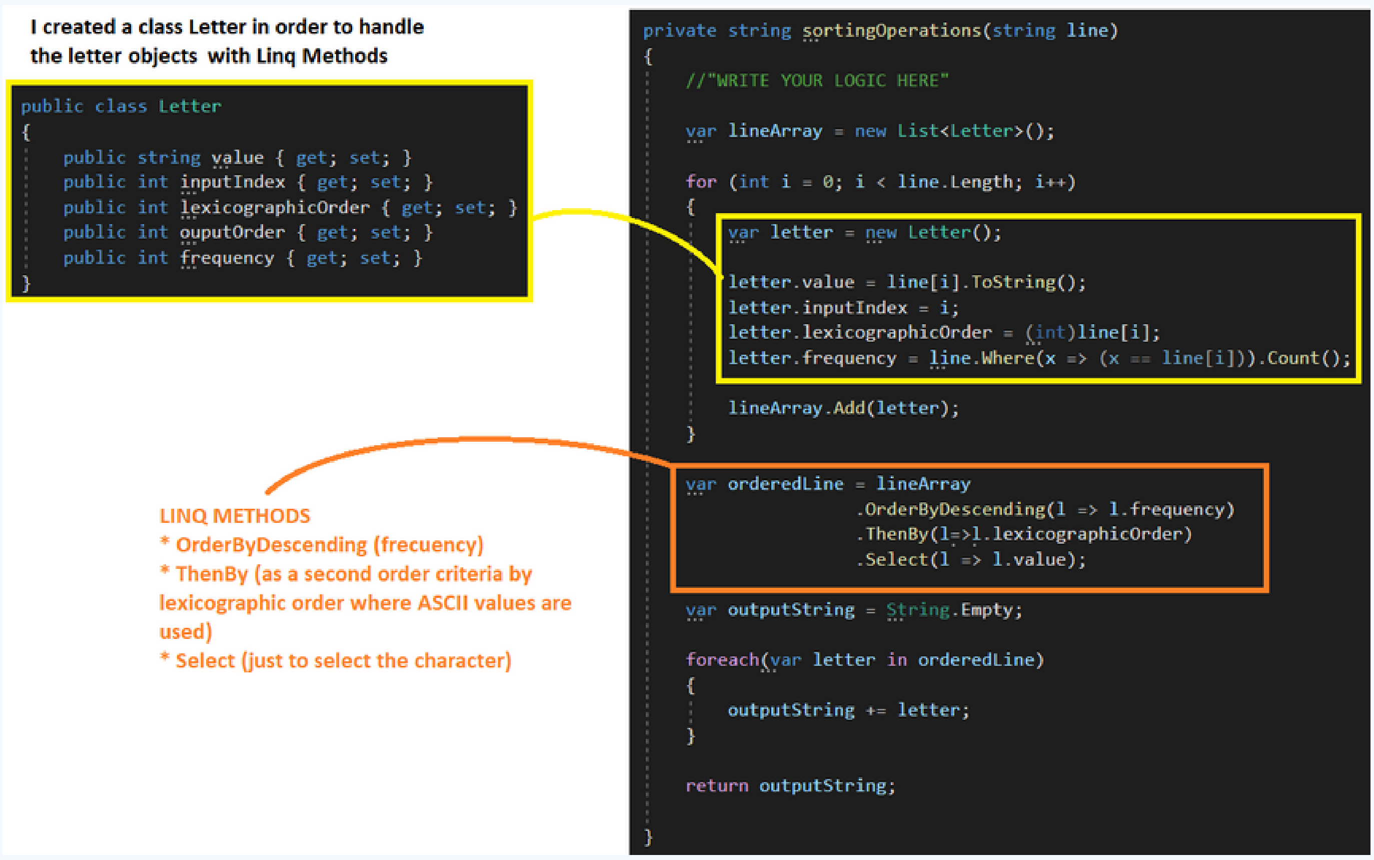
```csharp
private string sortingOperations(string line)
{
    //"WRITE YOUR LOGIC HERE"

    var lineArray = new List<Letter>();

    for (int i = 0; i < line.Length; i++)
    {
        var letter = new Letter();

        letter.value = line[i].ToString();
        letter.inputIndex = i;
        letter.lexicographicOrder = (int)line[i];
        letter.frequency = line.Where(x => (x == line[i])).Count();

        lineArray.Add(letter);
    }

    var orderedLine = lineArray
                      .OrderByDescending(l => l.frequency)
                      .ThenBy(l=>l.lexicographicOrder)
                      .Select(l => l.value);

    var outputString = String.Empty;

    foreach(var letter in orderedLine)
    {
        outputString += letter;
    }

    return outputString;

}
```

CLASS LETTER



```csharp
public class Letter
{
    public string value { get; set; }
    public int inputIndex { get; set; }
    public int lexicographicOrder { get; set; }
    public int ouputOrder { get; set; }
    public int frequency { get; set; }
}
```

CODE NOTES



I created a class Letter in order to handle the letter objects with Linq Methods

```
public class Letter
{
    public string value { get; set; }
    public int inputIndex { get; set; }
    public int lexicographicOrder { get; set; }
    public int ouputOrder { get; set; }
    public int frequency { get; set; }
}
```

```
private string sortingOperations(string line)
{
    //"WRITE YOUR LOGIC HERE"

    var lineArray = new List<Letter>();

    for (int i = 0; i < line.Length; i++)
    {
        var letter = new Letter();

        letter.value = line[i].ToString();
        letter.inputIndex = i;
        letter.lexicographicOrder = (int)line[i];
        letter.frequency = line.Where(x => (x == line[i])).Count();

        lineArray.Add(letter);
    }

    var orderedLine = lineArray
                    .OrderByDescending(l => l.frequency)
                    .ThenBy(l=>l.lexicographicOrder)
                    .Select(l => l.value);

    var outputString = String.Empty;

    foreach(var letter in orderedLine)
    {
        outputString += letter;
    }

    return outputString;

}
```

LINQ METHODS
* OrderByDescending (frecuency)
* ThenBy (as a second order criteria by lexicographic order where ASCII values are used)
* Select (just to select the character)

LOGIC BEHIND:

1. SPLIT THE RAW TEXT BY LINES
2. CREATE A LETTER CLASS
3. SET EACH LETTER OBJECT VALUES
4. CREATE LIST OF LETTERS
5. LINQ METHODS
6. RETURN REARRANGED STRING

LINKS

CODE GITHUB REPOSITORY

HTTPS://GITHUB.COM/JJMAYORAGON/ENCORACHALLENGE

DEMO VIDEO:

HTTPS://GITHUB.COM/JJMAYORAGON/ENCORACHALLENGE/BLOB/CBDFE0A14AF53C140AD7F94C88686E56B8040386/CODE%20CHALLENGE.MP4

SCREENSHOTS:

HTTPS://GITHUB.COM/JJMAYORAGON/ENCORACHALLENGE

READ ME TEXT:

HTTPS://GITHUB.COM/JJMAYORAGON/ENCORACHALLENGE/BLOB/CBDFE0A14AF53C140AD7F94C88686E56B8040386/README.TEXT