**Documentation**

The solution consists of 3 projects.

**1. OldPhonePadTranslator**

A class library where main logic exists and translation occurs.

**Explanation**

|  |
| --- |
| static readonly string[][] numToCharMappings = [  [" "],  ["&", "'", "("],  ["A", "B", "C"],  ["D", "E", "F"],  ["G", "H", "I"],  ["J", "K", "L"],  ["M", "N", "O"],  ["P", "Q", "R", "S"],  ["T", "U", "V"],  ["W", "X", "Y", "Z"]  ]; |

A 2-dimension arrays to store mappings of key pad number to respective characters. Since most of input will be number and also repeat in numeric value, can simply use array.

|  |
| --- |
| public static String OldPhonePad(string input)  {  if (string.IsNullOrEmpty(input) || !input.EndsWith('#')) throw new InvalidDataException(); |

Validate if the input can be transformed. Assumes that input must not be empty and ends with ‘#’.

|  |
| --- |
| string pattern = @"(\d|\W)\1\*";  Regex regex = new Regex(pattern);  var matchCollection = regex.Matches(input);  if (matchCollection.Count == 0) throw new InvalidDataException(); |

Uses regular expression to separate input into groups of same keys. It will match any number or special character with one or more consecutive identical. E.g. 22 3344\*# => 22, 33, 44, \*, #  
If there is no matches, will consider as invalid input.

|  |
| --- |
| if (char.IsNumber(match.Value, 0))  {  int index = int.Parse(match.Value[0].ToString());  if (match.Value.Length <= numToCharMappings[index].Length)  {  result += numToCharMappings[index][match.Value.Length - 1];  } |

Input group can be numbers or other characters. For numbers, can use mappings and directly translate to alphabetical or space. Also check if length of match value is within range of array index to prevent user-input fault.

|  |
| --- |
| else  {  result += numToCharMappings[index][(match.Value.Length % numToCharMappings[index].Length) - 1];  } |

If single digit is repeated multiple time, will assume that the character is repeated also.  
e.g. 2222 => A

|  |
| --- |
| // special cases  else if (match.Value == "\*")  {  result = result.Substring(0, result.Length - 1);  }  }  return result; |

If the input group is not number, need to consider as special cases and check the value and translate to respective output.

Finally, return the result message.

**2. OldPhonePad**

A console app to check the output of sample inputs.

**3. OldPhonePadTest**

Test project to validate the translator library with various test cases