Old Phone Keypad App Challenge

This challenge is aimed at creating an app modelling the old phone keypad. The keypad works by having each button corresponding to multiple letters. A single button press represents the first letter in the letters that are included in the button. Multiple presses run through the keypad to the next letter until all the presses are accounted for. Pauses allow for consecutive keypresses for input for instance; you must pause for a second to type two characters from the same button after other: “222 2 222#” -> “CAB”. A star (*****) is included as a backspace character for deleting a keypress. The hash (**#**) key is used as the send key. Zero (**0**) is used as a spacing character.

Keypad Features

The old phone keypad contains the following keys with their letters:

- 1: Special characters (.,$%&*@)
- 2: ABC
- 3: DEF
- 4: GHI
- 5: JKL
- 6: MNO
- 7: PQRS
- 8: TUV
- 9: WXYZ
- 0: Space key
- *: Backspace (removes the last keypad press)
- #: Send (signifies the last keypress and for sending the input)

What Old Phone Keypad Keys Do

1. Spacing: Use input **0** to output a space in between characters
2. Backspace: Keypress deletion occurs after sending the input backspace key star **(***)
3. Send: Input hash **(#**) to signify the send key
4. Pausing: Input space (blank) to signify a pause while making consecutive keypresses from the same key. For example, pressing **2 once** gives **A**, **twice** gives **B**, and **three times** gives **C**. Pressing **2 four times** moves back to letter **A**

Design for Old Phone Pad App

This is how our old phone keypad app works:

- Receiving Input: Get input by reading input from the standard input stream
- Mapping Characters to Letters: A **Dictionary** with key value pairs is used to store letters for each of the keypad buttons
- Counter: A counter is used to count the number of keypresses for a particular button
- Pausing: A pause allows for consecutive keypresses on the same button
- Backspacing: Remove a keypress by sending star **(*)**
- Send: Use the hash key **(#)** as the submit/ enter key

Design Considerations

Some of the design issues to consider include the following:

- Error Checking: Checking whether the input is empty or malformed
- Unit testing: Using **Microsoft .net** testing framework for unit testing covering edge cases to ensure that the program functions work as expected
- Performance: Using the **StringBuilder** class for efficient string composition. Using key value dictionary for efficient searching
- Code Readability: The code is reduced into a single reusable function which allows for readability, testing and extensibility

Example Inputs and Outputs

The summary below shows a sample of expected inputs and outputs:

- “222 2 22#” => CAB
- “33#” => E
- “227*#” => B
- “4433555 555666#” => HELLO
- “8 88777444666*664#” => TURING