**These test cases aim to verify the behavior of the OldPhonePad method implementation**

Test Scenario: Single Digit Keypad Input

Test Case Name: Single Digit Keypad Input  
Objective: Verify if the method correctly returns the output when the same key (3) is pressed twice.

Test Steps:

1. Initialize the method with the input "33#" and call OldPhonePad("33#").
2. The method should identify that the key 3 corresponds to the letters D, E, and F.
3. Since the key 3 is pressed twice, the method should return the second character mapped to key 3, which is E.
4. The method should detect the # key, signaling the end of input and stopping further processing.
5. Verify the output is E.

Expected Result: The output should be "E".

Test Scenario: Mixed Digits Without Backspace

Test Case Name: Mixed Digits Without Backspace  
Objective: Test the method with a mix of digits and spaces to confirm correct mapping and sequence handling.

Test Steps:

1. Input the sequence "4433555 555666#" into the method and call OldPhonePad("4433555 555666#").
2. The method should map the keys and accumulate characters based on the number of presses:
3. 4 → G
4. 4 → H
5. 3 (pressed three times) → E
6. Continue this pattern until all characters are processed.
7. Spaces indicate a pause, so the same key can be used to type different characters.
8. The method should stop processing once the # key is encountered.
9. Verify that the output matches "HELLO".

Expected Result: The output should be "HELLO".

Test Scenario: Backspace Functionality

Test Case Name: Backspace Functionality  
Objective: Validate that the \* key removes the previous character from the output.

Test Steps:

1. Call the method with "227\*#" using OldPhonePad("227\*#").
2. The method should process the sequence:
3. 2 (twice) → C
4. 7 → P.
5. The \* character indicates a backspace, removing the last character (P).
6. The # character indicates the end of input, stopping further processing.
7. Verify the final output is "B".

Expected Result: The output should be "B".

Test Scenario: Complex Input with Pauses

Test Case Name: Complex Input with Pauses  
Objective: Verify the behavior when the input contains spaces and a backspace character.

Test Steps:

1. Call the method with "8 88777444666\*664#" using OldPhonePad("8 88777444666\*664#").
2. The method should parse:
3. 8 → T
4. Space → pause.
5. 887 → O
6. 7 (three times) → P
7. 4 → I
8. 666 → N.
9. \* Removes the last character (N), so the sequence continues with 66.
10. Verify that the output after processing is "TOOL".
11. The method stops processing when the # character is encountered.

Expected Result: The output should be "TOOL".

Test Scenario: No Input (Edge Case)

Test Case Name: No Input (Edge Case)  
Objective: Test the behavior when there is no digit input before the #.

Test Steps:

1. Call the method with "#" using OldPhonePad("#").
2. The method should recognize that there are no digits before the # and thus return an empty string.
3. The method stops processing as # indicates the end of the input.
4. Verify the output is an empty string ("").

Expected Result: The output should be an empty string ("").

Test Scenario: Multiple Letters from Same Key

Test Case Name: Multiple Letters from Same Key  
Objective: Verify that the method handles multiple letters from the same key correctly when there is a pause (space).

Test Steps:

1. Call the method with "666 6#" using OldPhonePad("666 6#").
2. The method should interpret:
3. 666 (three times) → M
4. Space indicates a pause.
5. 6 → N.
6. The method should stop processing when # is reached.
7. Verify the final output is "MN".

Expected Result: The output should be "MN".

Test Scenario: Continuous Backspace Characters

Test Case Name: Continuous Backspace Characters  
Objective: Ensure that consecutive backspace (\*) characters remove the intended letters as expected.

Test Steps:

1. Call the method with "777\*77\*#" using OldPhonePad("777\*77\*#").
2. The method should process:
3. 777 → R
4. \* Removes R.
5. 77 → Q
6. \* Removes Q.
7. The final character remaining should be the result.
8. Verify that the output is "P" after processing.

Expected Result: The output should be "P".