Assignment Part 2

Tcl program that accepts a list of real numbers (or integers).

1. Calculates and displays 'median', the 'average', and the 'sum' of the numbers.
2. Generates and displays a new list containing the numbers in descending sorted order.
3. Generates and displays 2 new lists
    - first list: contains only the numbers that have /both/ digits '1' and '5'
    - second list: contains the numbers with /either/ digits '1' or '5'

This program folder contains 3 files including:

- **partTwo.tcl**: source code for part 2
- **README.md**: a guid to help user compile program in terminal, details of solution implementation, as well as test cases that have been performed and tested.
- **README.pdf**: a pdf version of README.md, for convenience

Prerequisites

Ensure to have `tclsh` installed on your development machine.

Execution

In order to execute the program, perform the `tclsh partTwo.tcl` command.

Discussion of solution implementation

The `tcl_precision` builtin variable is set to 6. This is done to keep clean the number of digits to retain when converting from float to string.

User input is validated for each entry. Each and every entry must be a number with an integer or a float. Numbers in the list are separated by newlines (pressing enter to enter the next element). If a period (.) is entered, the user would have indicated the end of list entries. If text (string) is entered, the user will get the following message: "Invalid input, please enter an integer:" and will be asked to reattempt. If a space is entered, the user will get the same previous message.

The *calculate_sum* procedure returns the sum of numbers in a list. The sum is calculated by stepping through the list, number by number, and iteratively adding those numbers and storing them in a variable.

The *calculate_avg* procedure returns the average of numbers in a list. The average is calculated by determining the length of the list (total count of numbers in the list), then it calls the *calculate_sum* procedure which returns the sum of numbers in the list. The procedure returns the results from dividing the sum by the length.

The *calculate_median* procedure returns the median of a list. It starts sorting the numbers in the list an ascending order. If the the total number of values in the list is odd, the procedure returns the middle value. If the total number of values in the list is even, the procedure returns the average of the two middle values.

A list of numbers can be sorted in a descending order using the `lsort` command with the `-decreasing` option. This will sort the numbers in the list from the largest to the smallest.

The *contains_1_or_5* procedure iterates through the list provided by the user and checks every number against the following regex patterns: `.*1.*` OR `.*5.*`. The `.` is used to match any character and the `*` to match the previous token as many times as possible. If either one of the conditions is satisfied, the number is added to a list that is later printed in the summary results. These patterns will check for numbers that include a 1 or 5 on the left of the number (eg. 10,57), the middle (213,351), or the right (481,135). If none of the numbers match the regex patterns, a message is printed instead saying "list is empty".

Similarly, the *contains_1_and_5* procedure iterates through the list provided by the user and checks every number against the following regex patterns: `.*1.*5.*` OR `.*5.*1.*`. These patterns will check for numbers that include 1 and 5 in the same number (eg 15, 51, 515, 151, 451). If none of the numbers match the regex patterns, a message is printed instead saying "list is empty".

Lastly, the program prints a summary of the results to the terminal.

List of test cases

calculate_sum procedure

1. All positive integers
    - Input: 3 23 45 65 2 9 60
    - Expected Output: 207
    - Status: ✅
2. All negative integers
    - Input: -4 -7 -9 -12 -100
    - Expected output: -132
    - Status: ✅
3. Zeroes
    - Input: 0 0 0
    - Expected output: 0
    - Status: ✅
4. Float numbers
    - Input: 5.3 1.935 -1.45 85 -99 0 -24 0.0
    - Expected output: -32.215
    - Status: ✅
5. Large numbers
    - Input: 10,000 12,566 99,537 100,000,000 45,950
    - Expected output: 1.00168e8
    - Status: ✅

6. One input
    - Input: -9
    - Expected output: -9
    - Status: ✅

calculate_average procedure

1. Positive integers
    - Input: 3 23 45 65 2 9 60
    - Expected Output: 29.5714
    - Status: ✅
2. Negative integers
    - Input: -4 -7 -9 -12 -100
    - Expected output: -26.4
    - Status: ✅
3. Zeroes
    - Input: 0 0 0
    - Expected output: 0
    - Status: ✅
4. Float numbers
    - Input: 5.3 1.935 -1.45 85 -99 0 -24 0.0
    - Expected output: -4.02688
    - Status: ✅
5. Large numbers
    - Input: 10,000 12,566 99,537 100,000,000 45,950
    - Expected output: 2.00336e7
    - Status: ✅
6. One input
    - Input: 10
    - Expected output: 10
    - Status: ✅

calculate_median procedure

1. Odd number of inputs
    - Input: 30 6 9 23 -5
    - Expected output: 9
    - Status: ✅

2. Even number of inputs

   - Input: 23 -100 23 45 -67 -4
   - Expected output:
   - Status: ✅

3. Positive numbers

   - Input: 14 99 82 29 67 1
   - Expected output: 48
   - Status: ✅

4. Negative numbers

   - Input: -100 -37 -2 -54
   - Expected output: -45.5
   - Status: ✅

5. Zeroes

   - Input: 0 0 0 0
   - Expected output: 0.0
   - Status: ✅

6. Float numbers

   - Input: 5.3 1.935 -1.45 85 -99 0 -24 0.0
   - Expected output: 0.0
   - Status: ✅

7. Large numbers

   - Input: 10,000 12,566 99,537 100,000,000 45,950
   - Expected output: 45950
   - Status: ✅

8. One input

   - Input: -7
   - Expected output: -7
   - Status: ✅

Printing list in a descending order

1. Positive integers
   - Input: 3 23 45 65 2 9 60
   - Expected Output: 65 60 45 23 9 3 2
   - Status: ✅

2. Negative integers
   - Input: -4 -7 -9 -12 -100
   - Expected output: -4 -7 -9 -12 -100
   - Status: ✅
3. Zeroes
   - Input: 0 0 0
   - Expected output: 0 0 0
   - Status: ✅
4. Float numbers
   - Input: 5.3 1.935 -1.45 85 -99 0 -24 0.0
   - Expected output: 85 5.3 1.935 0 0.0 -1.45 -24 -99
   - Status: ✅
5. Large numbers
   - Input: 10,000 12,566 99,537 1,000,000,000 45,950
   - Expected output: 1000000000 99537 45950 12566 10000
   - Status: ✅
6. One input
   - Input: -9
   - Expected output: -9
   - Status: ✅
7. Redundant numbers
   - Input: -25 -25 3 3 99 99 -7 -7
   - Expected output: 99 99 3 3 -7 -7 -25 -25
   - Status: ✅

Printing a list containing 1 or 5

1. Numbers with 1 or 5 on the left side of number
   - Input: -10 57 23 76 -200
   - Expected output: -10 57
   - Status: ✅
2. Numbers with 1 or 5 in the middle of number
   - Input: -213 23 76 -200 -2 357
   - Expected output: -210 357
   - Status: ✅

3. Numbers with 1 or 5 at the end of number
    - Input: -15 201 85 30 11
    - Expected output: -15 201 85 -11
    - Status: ✅
4. Numbers with 1 and 5
    - Input: -15 51 501 195 -33
    - Expected output: -15 51 195 501
    - Status: ✅
5. Empty list
    - Input: 23 43 78 98 -6
    - Expected output: list is empty
    - Status: ✅

Printing a list containing both 1 and 5

1. Numbers with 1 and 5
    - Input: -15 51 501 195 -33
    - Expected output: -15 51 195 501
    - Status: ✅
2. Numbers with 1 or 5
    - Input: -10 57 -5 12 -210
    - Expected output: list is empty
    - Status: ✅
3. Empty list
    - Input: 23 43 78 98 -6 5
    - Expected output: list is empty
    - Status: ✅