Comprehensive String Analyzer

1. Research (*Problem Identification*)

Text processing is essential across fields such as data analysis, NLP, and user interface design. A gap was identified in existing tools lacking a single solution that combines simplicity and versatility for educational purposes. The program aims to fulfil the need for an all-in-one string manipulation tool covering common operations like counting characters and words, case conversion, word length analysis, palindrome checking, and sorting.

Importance of Addressing the Problem:

- Centralized text manipulation functions simplify operations that are often spread across multiple functions or libraries.
- This program offers value as a learning tool for beginner programmers, enhancing foundational knowledge in string handling.

Researched credible sources such as:

- C Standard Library documentation for understanding function implementations.
- Academic resources on text processing methods and efficiencies.
- Case studies showing text processing needs in software applications.

2. Analyze (*Solutions & Benefits*)

Analysis of the problem focused on understanding user requirements for text manipulation functionality and identifying common challenges in string processing, such as case conversion, sorting, and removal of specific characters (digits or letters). The program requirements were clarified through group discussions, leading to a detailed breakdown of functionality:

- Character and word counting
- Case conversion
- Palindrome detection
- Reversal and sorting of characters
- Digit and letter removal

Our analysis included reviewing how each operation could be achieved with minimal computational complexity and how each function could independently contribute to an educational programming tool.

3. Ideate *(Program Functionality)*

Solution Development: Multiple approaches were considered for implementing each function, with a focus on efficiency and clarity. For example:

- Sorting: Alternatives like quicksort and bubble sort were evaluated, with bubble sort selected for simplicity and suitability for small strings.
- Case Conversion: Used standard library functions toupper() and tolower() for consistency.
- Recursive vs. Iterative Implementations: Functions like counting characters and checking palindromes were developed both recursively and iteratively to optimize speed and memory use.

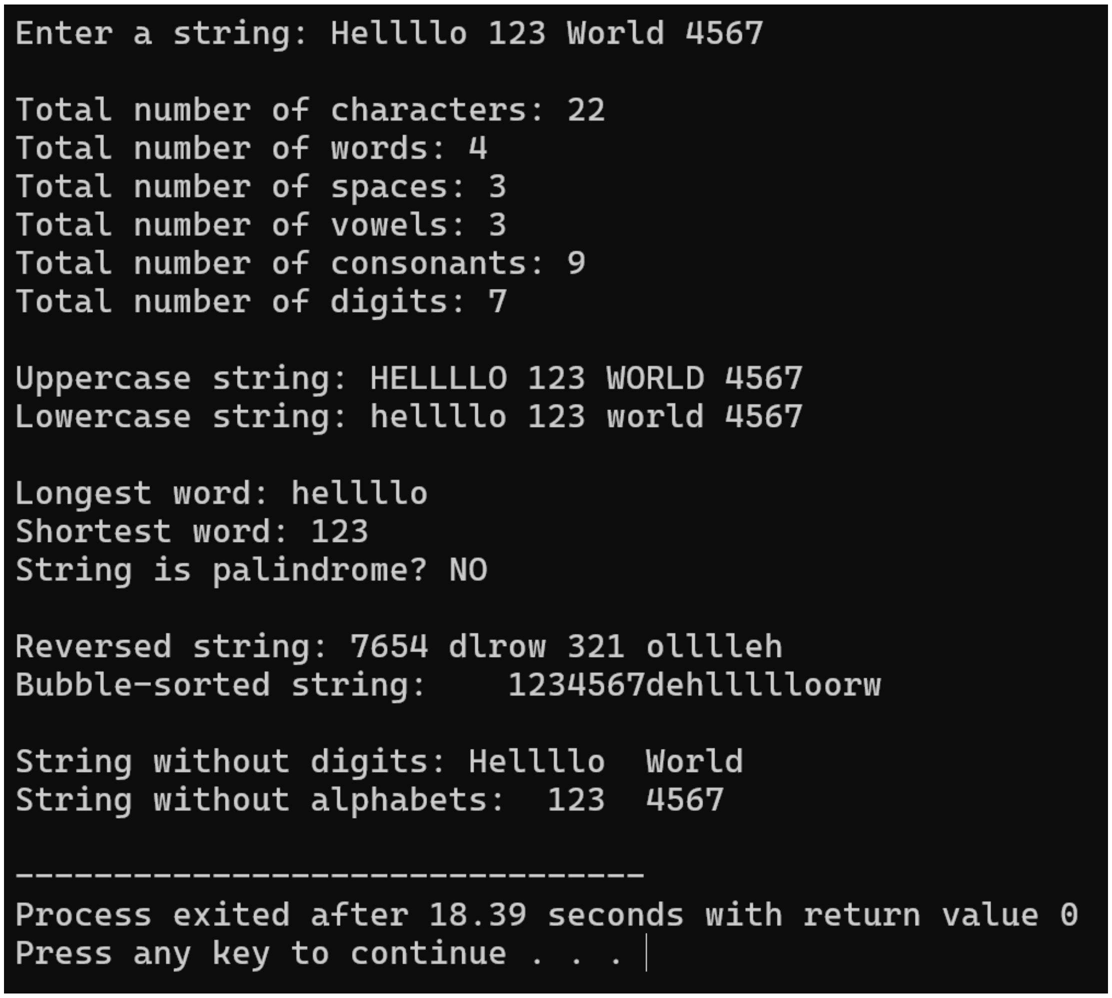
This ideation process led to a unique and innovative solution that combined multiple functionalities in one program, making it versatile for diverse text processing needs.

4. Build *(Error Handling)*

Assembly and Development: Each function was developed with meticulous attention to detail, ensuring compatibility and correct handling of edge cases, such as:

- Empty strings and single-character strings
- Strings with a mix of letters, numbers, and special characters
- Case sensitivity in palindrome detection

Functions were modularly designed and tested individually to avoid errors, allowing easy debugging and future scalability. The execute() function orchestrates the calls to each feature, ensuring cohesive program execution.



```
Enter a string: Hellllo 123 World 4567

Total number of characters: 22
Total number of words: 4
Total number of spaces: 3
Total number of vowels: 3
Total number of consonants: 9
Total number of digits: 7

Uppercase string: HELLLLO 123 WORLD 4567
Lowercase string: hellllo 123 world 4567

Longest word: hellllo
Shortest word: 123
String is palindrome? NO

Reversed string: 7654 dlrow 321 ollllеh
Bubble-sorted string:    1234567dehllllloorw

String without digits: Hellllo  World
String without alphabets:  123  4567

--------------------------------
Process exited after 18.39 seconds with return value 0
Press any key to continue . . .
```

5. Test *(Test Cases)*

Testing and Validation: Each functionality was rigorously tested across various scenarios, including:

- Edge cases: Single letters, empty strings, strings with no spaces or digits, etc.
- Functional cases: Testing uppercase/lowercase conversions, longest/shortest word identification, sorting accuracy, and palindrome detection.

The output consistently met or exceeded the desired objectives. We verified functionality by comparing results against manual calculations and ensured that the program handled large strings within the specified constraints (200 characters).

6. Implement (Deployment Applications)

Potential Use Cases: This program can be implemented in several contexts, including:

- Educational Institutions: Ideal for teaching basic programming concepts and text manipulation.
- Tech Companies: Could be integrated into larger applications requiring built-in string processing capabilities.
- NLP and Data Processing: Useful in pre-processing text data for machine learning models or analysis.

Industries:

- **EdTech companies** could use this as part of beginner coding courses.
- **Data Science firms** could implement its string functions to preprocess datasets.
- **Software Development agencies** might use this utility for backend operations where text processing is needed.

7. References:

- https://www.geeksforgeeks.org/bubble-sort-algorithm/
- https://www.geeksforgeeks.org/program-count-vowels-string-iterative-recursive/
- https://www.codesansar.com/c-programming-examples/longest-word-from-given-sentence.htm
- https://www.codesansar.com/c-programming-examples/shortest-word-from-given-sentence.htm
- https://www.w3schools.com/c/c_strings.php
- https://www.w3schools.com/c/c_ref_string.php
- https://www.tutorialspoint.com/c_standard_library/ctype_h.htm
- https://www.geeksforgeeks.org/sleep-function-in-c/
- https://www.freeformatter.com/string-utilities.html

8. Conclusion and Future Scope

Conclusion: The Comprehensive String Manipulation Program provides a robust solution for common text processing tasks, making it a valuable resource for both educational and practical programming applications. Its modularity and versatility allow it to fit various use cases, from educational tools to preprocessing utilities for data analysis.

Future Scope: Potential improvements include expanding the program to handle multi-language text, adding support for advanced text analysis functions like stemming or tokenization, and enhancing error handling for even broader input compatibility.

9. *Publish*

https://github.com/Isha6831/Comprehensive-String-Analyzer

10. Team Cast:

- **Parth R. Chaudhari** (Lead Developer & Presentation)
- **Rohan H. Mali** (Presentation. Assistant Coder)
- **Anurag Patil** (Conceptualization & Presentation & Assistant Developer)
- **Isha S. Patil** (Publisher & Assistant Developer & Presentation)

Thank You!