***ACTIVITY NO. 6***

#### **Group Members:**

• **Anurag Milind Patil (B24CE1047)**  
(Conceptualization & Assistant Developer)

• **Rohan Hemant Mali (B24CE1048)**  
(Presentation, Assistant Coder)

• **Isha Sandeep Patil (B24CE1049)**  
(Publisher & Assistant Developer & Presentation)

• **Parth R. Chaudhari (B24CE1050)**  
(Lead Developer & Presentation)

#### **Problem Statement:**

Develop a C program to calculate the sum of the first 10 numbers using recursive function call and pointers.

The program should define a recursive function that takes a pointer to an integer as an argument and returns the sum of first 10 numbers.

Within the function recursion should be used  
to compute the sum iteratively.

Finally the program should display  
the calculated sum.

***COMPREHENSIVE STRING ANALYSER***

#### **3. 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.

#### **4. 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.**

#### **5. Ideate (Program Functionality)**

The program was developed with a focus on efficiency and clarity, incorporating key concepts like recursion, iteration, string manipulation, and character classification. Highlights of the ideation process include:

#### **String Manipulation**: Utilized functions such as **strcpy()** and custom implementations to modify and process strings (e.g., reversing and sorting).

#### **Character Classification**: Leveraged standard library functions like **isalpha(), isdigit(), and isspace()** to analyse and filter string content.

#### **Input Handling**: Used **fgets()** to safely read strings, ensuring proper memory management and handling of newline characters.

#### **Case Conversion**: Employed **toupper() & tolower()** to change the string's case effectively.

#### **Sorting**: **Bubble sort** was selected for simplicity and ease of implementation, given the typically small size of input strings.

#### **Recursion**: Functions such as counting characters and checking for palindromes demonstrated elegant recursive implementations.

#### **This versatile solution integrates diverse functionalities into a single program, making it a practical tool for text manipulation & analysis.**

#### **6. Build (Code Overview)**

This program demonstrates a variety of string operations, providing an interactive way to understand text analysis and manipulation.

**What the User Needs to Do:**

1. **Enter a string** of their choice when prompted.
2. **Observe the detailed analysis** and modifications performed on the string by the program.

**What the Program Does:**

* **Analyzes the String**: Counts characters, words, spaces, vowels, consonants, and digits.
* **Manipulates the String**: Converts it to uppercase and lowercase, reverses it, and sorts the characters in ascending order.
* **Finds Words**: Identifies the longest and shortest words.
* **Checks for Palindromes**: Determines if the string is a palindrome.
* **Removes Characters**: Creates modified versions of the string by removing digits or alphabetic characters.

**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.**

**7*. Publish***

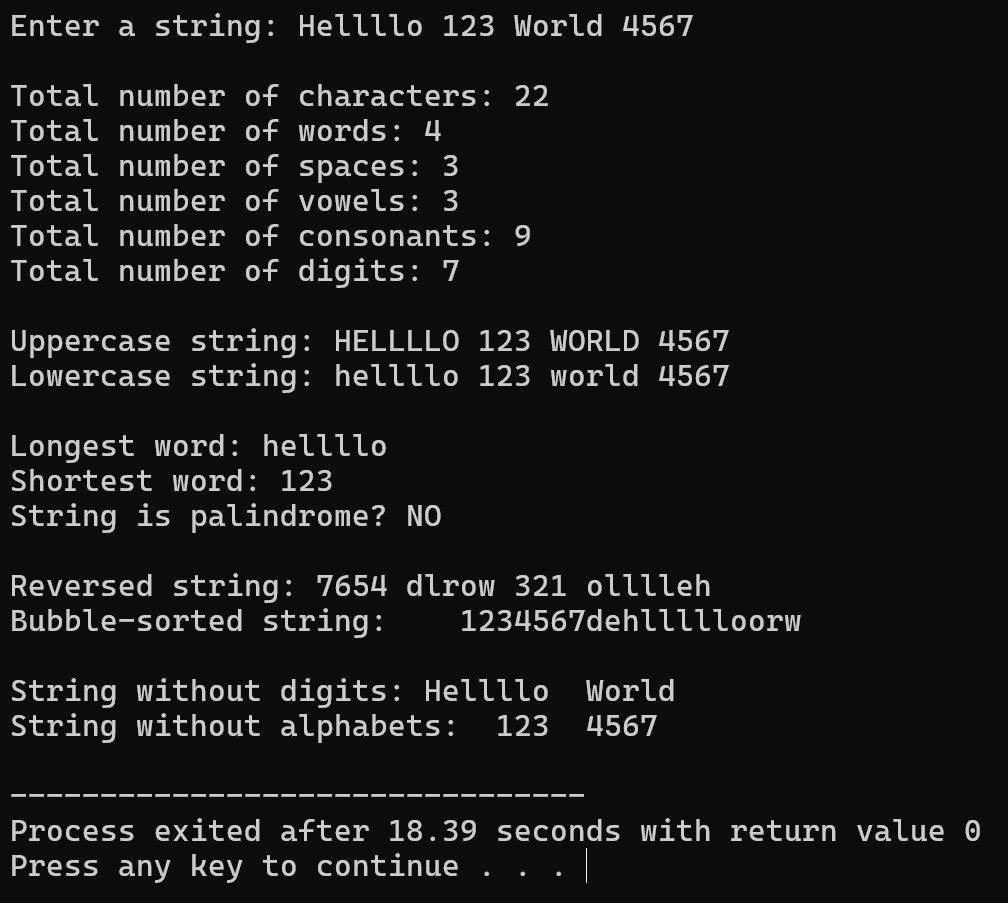
[**https://github.com/Isha6831/Comprehensive-String-Analyzer**](https://github.com/Isha6831/Comprehensive-String-Analyzer)

**8. *Test (****Test Cases Handling****)***

**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.



#### **9. 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.**

***10. Conclusion & 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.

#### **11. 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>

**Thank You!**