1. **String Reversal**: Write a C# program to reverse a given string without using any built-in functions.
2. **Anagram Detection**: Create a function to check if two strings are anagrams of each other.
3. **Palindrome Check**: Write a program to determine if a given string is a palindrome or not.
4. **String Compression**: Implement a function to perform basic string compression using the counts of repeated characters.
5. **String Concatenation**: Compare the performance of string concatenation using the **+** operator vs. **StringBuilder**.
6. **Substring Search**: Write a function to find all occurrences of a substring within a given string.
7. **String Tokenization**: Implement a function to tokenize a given string based on a delimiter.
8. **Longest Substring Without Repeating Characters**: Find the length of the longest substring without repeating characters in a given string.
9. **String Rotation**: Write a program to check if one string is a rotation of another.
10. **Count Vowels and Consonants**: Create a function to count the number of vowels and consonants in a given string.
11. **String Palindrome Permutation**: Determine if a given string can be rearranged into a palindrome.
12. **String Matching Algorithm**: Implement a simple pattern matching algorithm like Naive Pattern Searching or Rabin-Karp Algorithm.
13. **String Formatting**: Write a function to format a string into a specific pattern (e.g., phone number format, date format).
14. **Duplicate Characters**: Find and print the duplicate characters in a given string.
15. **String Comparison**: Compare two strings lexicographically without using any built-in functions.
16. **Word Frequency Counter**: Create a program to count the frequency of each word in a given string.
17. **String Rotation Check**: Write a function to check if one string is a rotation of another using only one call to **Substring**.
18. **String Permutations**: Generate all permutations of a given string.
19. **String Truncation**: Write a function to truncate a given string to a specified length and append "..." if it exceeds the length.

**String Manipulation:**

1. Write a C# program to concatenate two strings using the **+** operator.
2. Write a C# program to find the length of a string in C#.
3. Write a program to convert a string to uppercase and lowercase in C#?
4. Write a program to reverse a given string without using any built-in functions.
5. Implement a method to check if a given string, is a palindrome.
6. Write a C# method to count the number of occurrences of a specific character in a string.

**StringBuilder Usage:**

1. Describe the StringBuilder class in C#. What are its main features and advantages?
2. Compare string concatenation using StringBuilder versus using the **+** operator in terms of performance and memory usage.
3. Write a C# program to dynamically construct a large string by appending various components using StringBuilder.
4. Explain how StringBuilder helps in mitigating the performance overhead associated with string immutability.
5. Discuss scenarios where using StringBuilder is more appropriate than regular string concatenation methods.
6. Demonstrate the usage of StringBuilder for building XML or HTML documents in C#.
7. Write a program to demonstrate the efficiency of StringBuilder in string manipulation tasks involving loops or repeated operations.
8. **Advanced Topics:**
9. Implement a function to find all occurrences of a substring within a given string using StringBuilder.
10. Discuss the performance implications of string concatenation operations in multi-threaded environments. How can StringBuilder help in such scenarios?
11. Write a program to generate CSV (Comma-Separated Values) or TSV (Tab-Separated Values) files from structured data using StringBuilder.
12. Explain the concept of string interpolation and how it differs from traditional string concatenation methods.
13. Discuss the importance of memory management and resource optimization when working with strings in C#. How can StringBuilder contribute to efficient memory usage?