Assignment 7

Topics: Recursion and Basics of Strings

Section A7.1: [Recursion Basics]

A7.1a: Write a recursive function to calculate the factorial of a given number. Call the function in main() and print the result.

- **Hint**: The base case is when the number is 0 or 1 (0! = 1 and 1! = 1). For other values, return n * factorial(n-1).

A7.1b: Write a recursive function to calculate the sum of digits of an integer. Use the function in main() to find the sum of digits for a user-provided number.

- **Hint**: Base case: when the number becomes 0. Recursion: return (n % 10) + sumOfDigits(n / 10).

A7.1c: Write a recursive function to calculate the nth Fibonacci number. Call this function in main() to display the Fibonacci sequence up to n.

- **Hint**: Base cases: n = 0 (return 0), n = 1 (return 1). Recursion: fibonacci(n-1) + fibonacci(n-2).

A7.1d [Bonus]: Write a recursive function to reverse the digits of an integer. Call the function in main() to reverse a user-entered number.

- **Hint**: Use recursion to extract digits from the number, and reconstruct the reversed number.

Section A7.2: [Advanced Recursion Problems]

A7.2a: Write a recursive function to find the Greatest Common Divisor (GCD) of two numbers using the Euclidean algorithm. Call the function in main() to find the GCD of two user-provided integers.

- **Hint**: Base case: when one of the numbers becomes 0. Recursion: gcd(a, b) = gcd(b, a % b).

A7.2b: Implement a recursive function to solve the Tower of Hanoi problem. The function should print the moves required to transfer disks from the source peg to the destination peg.

- **Hint**: Base case: Move one disk directly. Recursion: Move n-1 disks to an auxiliary peg, move the last disk, then move n-1 disks to the destination peg.

A7.2c [Bonus]: Write a recursive function to generate all permutations of a string. Call this function in main() to display all permutations of a user-provided string.

- **Hint**: Swap characters at each position, then recursively generate permutations for the rest of the string.

A7.2d: Recursion for Pascal's Triangle

Write a recursive function that prints the nth row of Pascal's Triangle. Pascal's Triangle is constructed using the binomial coefficient formula:

$$\left[C(n,\, k) = \frac{n!}{k!\,(n-k)!} \right]$$

Call the function in main() to display the nth row based on user input.

- **Hint**: Use the recursive relation: $[\,C(n,\, k) = C(n-1,\, k-1) + C(n-1,\, k)]$with base cases when k == 0 or k == n.

A7.2e [Bonus]: Recurrence for Catalan Numbers

Write a recursive function to compute the nth Catalan number. Catalan numbers follow the recurrence relation:

$$\left[C_n = \sum^{\cdot}_{\{i=0\}^{\{n-1\}C}_{i}} C_{\{n-i-1\}} \right]$$

Call the function in main() and print the nth Catalan number.

- **Hint**: Base case: $C_0 = 1$. Recursion: Use the sum formula recursively to compute higher Catalan numbers.

Section A7.3: [Basic String Operations]

A7.3a: Write a program that calculates the length of a string without using the standard strlen() function. Implement your own function to count the characters in the string.

- **Hint**: Traverse the string character by character until you encounter the null character '\0'.

A7.3b: Write a function that copies one string to another without using strcpy(). Implement this function and call it in main() to copy a user-provided string.

- **Hint**: Traverse both strings character by character, copying from the source to the destination.

A7.3c: Write a program that compares two strings lexicographically without using strcmp(). The program should return 0 if the strings are equal, a positive number if the first string is greater, and a negative number if the second string is greater.

- **Hint**: Compare the strings character by character. If characters differ, return the difference; otherwise, continue until the end of the strings.

A7.3d [Bonus]: Write a program that removes all vowels from a given string. The program should take a string as input, modify it to remove the vowels, and then print the result.

- **Hint**: Traverse the string, checking each character. Skip adding vowels (a, e, i, o, u) to the result.

Section A7.4: [String Manipulation using Pointers]

A7.4a: Write a program that reverses a string using pointers. The function should take a pointer to the string and reverse it in place.

- **Hint**: Use two pointers, one starting at the beginning and the other at the end of the string. Swap the characters and move the pointers toward each other until they meet.

A7.4b: Write a function that concatenates two strings using pointers without using strcat(). Implement this function and use it to concatenate two user-provided strings.

- **Hint**: Use a pointer to traverse the first string until the null terminator, then copy the second string starting at that position.

A7.4c: Write a program that counts the number of words in a string using pointers. Words are separated by spaces.

- **Hint**: Use a pointer to traverse the string, count spaces, and avoid multiple spaces between words.

A7.4d [Bonus]: Write a program that finds and prints the longest word in a string using pointers.

- **Hint**: Use two pointers to mark the start and end of each word. Keep track of the longest word by comparing lengths as you traverse the string.