Recursion

PCC104 - Prof. Rodrigo Silva

Leitura Recomendada

- Recursão `https://panda.ime.usp.br/pensepy/static/pensepy/12-Recursao/recursionsimpleptbr.html`

Problem Set

1. **Print 1 to n without using loops.**
   Write a recursive function to print numbers from 1 to n without using loops.
   Example:
   Input: n = 5
   Output: 1 2 3 4 5
   Test Case:
   Input: n = 7
   Expected Output: 1 2 3 4 5 6 7

2. **Print N to 1 without loop.**
   Write a recursive function to print numbers from N to 1 without using loops.
   Example:
   Input: N = 5
   Output: 5 4 3 2 1
   Test Case:
   Input: N = 8
   Expected Output: 8 7 6 5 4 3 2 1

3. **Mean of an Array using Recursion.**
   Create a recursive function that computes the mean of an array of numbers.
   Example:
   Input: arr = [1, 2, 3, 4, 5]
   Output: 3
   Test Case:
   Input: arr = [10, 20, 30, 40, 50]
   Expected Output: 30

4. **Sum of natural numbers using recursion.**
   Write a recursive function to compute the sum of the first n natural numbers.

Example:
Input: n = 5
Output: 15
Test Case:
Input: n = 10
Expected Output: 55

5. **Decimal to binary number using recursion.**
   Implement a recursive function to convert a decimal number to its binary equivalent.
   Example:
   Input: n = 10
   Output: 1010
   Test Case:
   Input: n = 15
   Expected Output: 1111

6. **Sum of array elements using recursion.**
   Write a recursive function to calculate the sum of all elements in an array.
   Example:
   Input: arr = [1, 2, 3, 4, 5]
   Output: 15
   Test Case:
   Input: arr = [10, 20, 30, 40]
   Expected Output: 100

7. **Print reverse of a string using recursion.**
   Write a recursive function to print the reverse of a given string.
   Example:
   Input: str = "hello"
   Output: "olleh"
   Test Case:
   Input: str = "recursion"
   Expected Output: "noisrucer"

8. **Program for length of a string using recursion.**
   Create a recursive function to find the length of a given string.
   Example:
   Input: str = "hello"
   Output: 5
   Test Case:
   Input: str = "recursion"
   Expected Output: 9

9. **Sum of digits of a number using recursion.**
   Write a recursive function to compute the sum of the digits of a given number.
   Example:

Input: n = 1234
Output: 10
Test Case:
Input: n = 9876
Expected Output: 30

10. **Tail recursion to calculate sum of array elements.**
Explain what tail recursion is, and write a tail-recursive function to calculate the sum of array elements.
Explanation: Tail recursion occurs when the recursive call is the last statement to be executed in the function. This allows for optimization where the compiler can reuse the same stack frame, improving efficiency.
Example:
Input: arr = [1, 2, 3, 4]
Output: 10
Test Case:
Input: arr = [5, 10, 15, 20]
Expected Output: 50

11. **Program to print first n Fibonacci Numbers.**
Write a recursive function to print the first n Fibonacci numbers.
Example:
Input: n = 5
Output: 0 1 1 2 3
Test Case:
Input: n = 7
Expected Output: 0 1 1 2 3 5 8

12. **Program for factorial of a number.**
Write a recursive function to compute the factorial of a given number.
Example:
Input: n = 5
Output: 120
Test Case:
Input: n = 7
Expected Output: 5040

13. **Recursive Programs to find Minimum and Maximum elements of an array.**
Write two recursive functions to find the minimum and maximum elements of a given array.
Example:
Input: arr = [1, 4, 3, -5, 10]
Output: Minimum = -5, Maximum = 10
Test Case:
Input: arr = [20, 15, 25, 5]
Expected Output: Minimum = 5, Maximum = 25

14. **Recursive function to check if a string is palindrome.**
Write a recursive function to check if a given string is a palindrome.
Example:
Input: str = "radar"
Output: True
Test Case:
Input: str = "hello"
Expected Output: False

15. **Print Fibonacci Series in reverse order using Recursion.**
Write a recursive function to print the Fibonacci series in reverse order.
Example:
Input: n = 5
Output: 3 2 1 1 0
Test Case:
Input: n = 7
Expected Output: 8 5 3 2 1 1 0

16. **Coin Change – Count Ways to Make Sum.**
Last Updated: 24 Aug, 2024
Given an integer array `coins[]` of size N representing different types of denominations and an integer `sum`, the task is to count all combinations of coins to make a given value `sum`. Assume that you have an infinite supply of each type of coin.

Examples:
Input: sum = 4, coins[] = {1, 2, 3}
Output: 4
Explanation: There are four solutions: {1, 1, 1, 1}, {1, 1, 2}, {2, 2}, and {1, 3}.

Input: sum = 10, coins[] = {2, 5, 3, 6}
Output: 5
Explanation: There are five solutions: {2, 2, 2, 2, 2}, {2, 2, 3, 3}, {2, 2, 6}, {2, 3, 5}, and {5, 5}.

Input: sum = 10, coins[] = {10}
Output: 1
Explanation: The only solution is to pick 1 coin of value 10.

Input: sum = 5, coins[] = {4}
Output: 0
Explanation: We cannot make sum 5 with the given coins.

Input: sum = 5, coins[] = {1, 2}
Expected Output: 3
Explanation: The three solutions are {1, 1, 1, 1, 1}, {1, 2, 2}, and {2, 1, 1, 1}.

Input: sum = 7, coins[] = {2, 3, 6}
Expected Output: 2
Explanation: The two solutions are {2, 2, 3} and {1, 3, 3}.

Input: sum = 8, coins[] = {2, 4, 6}
Expected Output: 3
Explanation: The three solutions are {2, 2, 2, 2}, {2, 2, 4}, and {4, 4}.

17. **Binary Search using Recursion.**
Write a recursive function that implements the binary search algorithm to find the position of a target element in a sorted array. If the target element is not present, the function should return -1.

Algorithm Explanation: - Binary search works by repeatedly dividing the search interval in half. - If the value of the search key is less than the item in the middle of the interval, the search continues in the lower half. - If the value is greater, the search continues in the upper half. - This process continues until the value is found or the interval is empty.

Examples:
Input: arr = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, target = 6
Output: 5
Explanation: The element 6 is at index 5 (0-based indexing).

Input: arr = {1, 2, 3, 4, 5}, target = 10
Output: -1
Explanation: The element 10 is not present in the array.

Input: arr = {10, 20, 30, 40, 50}, target = 30
Expected Output: 2
Explanation: The element 30 is at index 2.

Input: arr = {5, 15, 25, 35, 45, 55}, target = 45
Expected Output: 4
Explanation: The element 45 is at index 4.

Input: arr = {7, 14, 21, 28, 35, 42}, target = 50
Expected Output: -1
Explanation: The element 50 is not present in the array.

Input: arr = {2, 3, 5, 7, 11, 13, 17, 19}, target = 7
Expected Output: 3
Explanation: The element 7 is at index 3.