Functions Problems

1. Find the maximum of two numbers

c

CopyEdit

```
#include <stdio.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

int main() {
    int x = 10, y = 20;
    printf("Maximum: %d\n", max(x, y));
    return 0;
}
```

Explanation: Uses the ternary operator to return the larger number.

---

2. Sum of two numbers

c

CopyEdit

```
#include <stdio.h>

int sum(int a, int b) {
    return a + b;
}

int main() {
    printf("Sum: %d\n", sum(5, 10));
    return 0;
}
```

Explanation: Simply adds two numbers and returns the result.

---

3. Check if a number is even or odd

c
CopyEdit
```
#include <stdio.h>

void checkEvenOdd(int n) {
    if (n % 2 == 0)
        printf("%d is Even\n", n);
    else
        printf("%d is Odd\n", n);
}

int main() {
    checkEvenOdd(7);
    return 0;
}
```

Explanation: Uses modulus operator % to determine if n is even or odd.

---

4. Swap two numbers using pointers

c
CopyEdit
```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x = 5, y = 10;
    swap(&x, &y);
    printf("After Swap: x = %d, y = %d\n", x, y);
    return 0;
}
```

Explanation: Uses pointers to modify values directly in memory.

---

5. Factorial using iteration

c
CopyEdit

```
#include <stdio.h>

int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++)
        fact *= i;
    return fact;
}

int main() {
    printf("Factorial: %d\n", factorial(5));
    return 0;
}
```

Explanation: Multiplies numbers from 1 to n using a loop.

Recursion Problems

6. Factorial using recursion

c
CopyEdit

```
#include <stdio.h>
```

```
int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}

int main() {
    printf("Factorial: %d\n", factorial(5));
    return 0;
}
```

Explanation: Calls itself until `n == 0`, then returns 1.

---

7. Fibonacci sequence (nth term)

c

CopyEdit

```
#include <stdio.h>

int fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    printf("Fibonacci(6): %d\n", fibonacci(6));
    return 0;
}
```

Explanation: Calls itself to compute the sum of two preceding numbers.

---

8. Sum of digits using recursion

c
CopyEdit
```c
#include <stdio.h>

int sumDigits(int n) {
    if (n == 0) return 0;
    return (n % 10) + sumDigits(n / 10);
}

int main() {
    printf("Sum of digits: %d\n", sumDigits(1234));
    return 0;
}
```

Explanation: Extracts last digit and adds it recursively.

---

9. Reverse a number using recursion

c
CopyEdit
```c
#include <stdio.h>

int reverse(int n, int rev) {
    if (n == 0) return rev;
    return reverse(n / 10, rev * 10 + n % 10);
}

int main() {
    printf("Reversed Number: %d\n", reverse(1234, 0));
    return 0;
}
```

Explanation: Keeps building the reversed number by shifting digits.

---

10. Power function using recursion (x^y)

c

CopyEdit

```
#include <stdio.h>

int power(int x, int y) {
    if (y == 0) return 1;
    return x * power(x, y - 1);
}

int main() {
    printf("Power: %d\n", power(2, 3));
    return 0;
}
```

Explanation: Multiplies x by itself y times.

11. Function to find GCD using recursion

c

CopyEdit

```
#include <stdio.h>

int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int main() {
    printf("GCD: %d\n", gcd(48, 18));
    return 0;
}
```

Explanation: Uses Euclidean algorithm where `GCD(a, b) = GCD(b, a % b)`. Stops when `b == 0`.

---

12. Function to find LCM using GCD

c
CopyEdit
```
#include <stdio.h>

int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int lcm(int a, int b) {
    return (a * b) / gcd(a, b);
}

int main() {
    printf("LCM: %d\n", lcm(12, 18));
    return 0;
}
```

Explanation: Uses the formula `LCM(a, b) = (a * b) / GCD(a, b)`.

---

13. Function to check if a number is prime

c
CopyEdit
```
#include <stdio.h>

int isPrime(int n, int i) {
    if (n <= 2) return (n == 2) ? 1 : 0;
    if (n % i == 0) return 0;
    if (i * i > n) return 1;
    return isPrime(n, i + 1);
}
```

```
int main() {
    printf("Is Prime: %d\n", isPrime(29, 2));
    return 0;
}
```

Explanation: Checks divisibility up to `sqrt(n)`, returns 1 if no divisors found.

---

14. Function to find sum of first n natural numbers

c
CopyEdit
```
#include <stdio.h>

int sumNatural(int n) {
    if (n == 0) return 0;
    return n + sumNatural(n - 1);
}

int main() {
    printf("Sum: %d\n", sumNatural(5));
    return 0;
}
```

Explanation: Uses formula `n + (n-1) + (n-2) ... + 1`.

---

15. Function to reverse a string using recursion

c
CopyEdit
```
#include <stdio.h>

void reverseString(char str[], int start, int end) {
    if (start >= end) return;
    char temp = str[start];
    str[start] = str[end];
    str[end] = temp;
    reverseString(str, start + 1, end - 1);
}

int main() {
    char str[] = "hello";
    reverseString(str, 0, 4);
    printf("Reversed: %s\n", str);
    return 0;
}
```

Explanation: Swaps characters recursively until the middle is reached.

---

16. Function to check if a string is a palindrome

c
CopyEdit
```
#include <stdio.h>

int isPalindrome(char str[], int start, int end) {
    if (start >= end) return 1;
    if (str[start] != str[end]) return 0;
    return isPalindrome(str, start + 1, end - 1);
}

int main() {
    char str[] = "madam";
    printf("Palindrome: %d\n", isPalindrome(str, 0, 4));
    return 0;
}
```

Explanation: Compares first and last characters, moving inward recursively.

---

17. Function to print numbers from n to 1

c
CopyEdit

```c
#include <stdio.h>

void printNumbers(int n) {
    if (n == 0) return;
    printf("%d ", n);
    printNumbers(n - 1);
}

int main() {
    printNumbers(5);
    return 0;
}
```

Explanation: Prints n and calls itself with `n-1`.

---

18. Function to convert decimal to binary

c
CopyEdit

```c
#include <stdio.h>

void decimalToBinary(int n) {
    if (n == 0) return;
    decimalToBinary(n / 2);
    printf("%d", n % 2);
}

int main() {
    decimalToBinary(10);
    return 0;
}
```

Explanation: Recursively divides n by 2 and prints remainder.

---

19. Function to find nth term of an arithmetic sequence

c
CopyEdit
```
#include <stdio.h>

int nthTerm(int a, int d, int n) {
    return a + (n - 1) * d;
}

int main() {
    printf("Nth Term: %d\n", nthTerm(2, 3, 5));
    return 0;
}
```

Explanation: Uses formula `a + (n-1) * d`.

---

20. Function to find nth term of a geometric sequence

c
CopyEdit
```
#include <stdio.h>

int nthTermGeo(int a, int r, int n) {
    if (n == 1) return a;
    return nthTermGeo(a * r, r, n - 1);
}

int main() {
    printf("Nth Term: %d\n", nthTermGeo(2, 3, 4));
    return 0;
}
```

Explanation: Uses formula a `* r^(n-1)` recursively.

Recursion Problems

21. Count digits in a number

c
CopyEdit
```
#include <stdio.h>

int countDigits(int n) {
    if (n == 0) return 0;
    return 1 + countDigits(n / 10);
}

int main() {
    printf("Digits: %d\n", countDigits(1234));
    return 0;
}
```

Explanation: Divides n by `10` recursively, counting each step.

---

22. Check if an array is sorted

c
CopyEdit
```
#include <stdio.h>

int isSorted(int arr[], int n) {
    if (n == 1) return 1;
    if (arr[n - 1] < arr[n - 2]) return 0;
    return isSorted(arr, n - 1);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printf("Is Sorted: %d\n", isSorted(arr, 5));
    return 0;
}
```

Explanation: Compares adjacent elements recursively.

---

23. Calculate nCr using recursion

c
CopyEdit
```
#include <stdio.h>

int nCr(int n, int r) {
    if (r == 0 || n == r) return 1;
    return nCr(n - 1, r - 1) + nCr(n - 1, r);
}

int main() {
    printf("nCr: %d\n", nCr(5, 2));
    return 0;
}
```

Explanation: Uses Pascal's triangle formula.

---

24. Solve Tower of Hanoi

c
CopyEdit
```
#include <stdio.h>
```

```
void hanoi(int n, char from, char to, char aux) {
    if (n == 0) return;
    hanoi(n - 1, from, aux, to);
    printf("Move disk %d from %c to %c\n", n, from, to);
    hanoi(n - 1, aux, to, from);
}

int main() {
    hanoi(3, 'A', 'C', 'B');
    return 0;
}
```

Explanation: Moves n-1 disks to auxiliary peg, then moves largest, then moves n-1 disks again.

25. Check if an array is sorted using recursion

c
CopyEdit

```
#include <stdio.h>

int isSorted(int arr[], int n) {
    if (n == 1) return 1; // A single element array is always
sorted
    if (arr[n - 1] < arr[n - 2]) return 0; // If the current
element is smaller than the previous one, return false
    return isSorted(arr, n - 1); // Check the rest of the array
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printf("%s\n", isSorted(arr, 5) ? "Sorted" : "Not Sorted");
    return 0;
}
```

Explanation: Recursively checks if each element is greater than or equal to the previous one.

---

26. Print numbers from 1 to N using recursion

c
CopyEdit
```
#include <stdio.h>

void printNumbers(int n) {
    if (n == 0) return;
    printNumbers(n - 1);
    printf("%d ", n);
}

int main() {
    printNumbers(10);
    return 0;
}
```

Explanation: Calls itself with `n-1` first, ensuring numbers are printed in ascending order.

---

27. Print numbers from N to 1 using recursion

c
CopyEdit
```
#include <stdio.h>

void printNumbersReverse(int n) {
    if (n == 0) return;
    printf("%d ", n);
    printNumbersReverse(n - 1);
}
```

```
int main() {
    printNumbersReverse(10);
    return 0;
}
```

Explanation: Prints the number first, then calls itself with `n-1`, ensuring descending order.

---

28. Find the Greatest Common Divisor (GCD) using recursion

c
CopyEdit

```
#include <stdio.h>

int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int main() {
    printf("GCD: %d\n", gcd(48, 18));
    return 0;
}
```

Explanation: Uses the **Euclidean algorithm**, which repeatedly reduces the problem using `a % b`.

---

29. Find the Least Common Multiple (LCM) using recursion

c
CopyEdit

```
#include <stdio.h>
```

```
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

int lcm(int a, int b) {
    return (a / gcd(a, b)) * b;
}

int main() {
    printf("LCM: %d\n", lcm(12, 15));
    return 0;
}
```

Explanation: Uses the formula `LCM(a, b) = (a * b) / GCD(a, b)`.

---

30. Count the number of digits in a number using recursion

c

CopyEdit

```
#include <stdio.h>

int countDigits(int n) {
    if (n == 0) return 0;
    return 1 + countDigits(n / 10);
}

int main() {
    printf("Number of digits: %d\n", countDigits(12345));
    return 0;
}
```

Explanation: Removes the last digit and counts +1 recursively.

31. Reverse an array using recursion

c
CopyEdit
```
#include <stdio.h>

void reverseArray(int arr[], int start, int end) {
    if (start >= end) return;
    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
    reverseArray(arr, start + 1, end - 1);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = 5;
    reverseArray(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Explanation: Swaps first and last elements recursively until the middle is reached.

---

32. Find sum of array elements using recursion

c
CopyEdit
```
#include <stdio.h>

int sumArray(int arr[], int n) {
    if (n == 0) return 0;
    return arr[n - 1] + sumArray(arr, n - 1);
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printf("Sum: %d\n", sumArray(arr, 5));
    return 0;
}
```

Explanation: Adds last element to sum of remaining elements recursively.

---

33. Find maximum element in an array using recursion

c

CopyEdit

```
#include <stdio.h>

int findMax(int arr[], int n) {
    if (n == 1) return arr[0];
    int maxRest = findMax(arr, n - 1);
    return (arr[n - 1] > maxRest) ? arr[n - 1] : maxRest;
}

int main() {
    int arr[] = {3, 1, 4, 1, 5, 9};
    printf("Max: %d\n", findMax(arr, 6));
    return 0;
}
```

Explanation: Compares last element with max of the rest recursively.

---

34. Find minimum element in an array using recursion

c

CopyEdit
```
#include <stdio.h>

int findMin(int arr[], int n) {
    if (n == 1) return arr[0];
    int minRest = findMin(arr, n - 1);
    return (arr[n - 1] < minRest) ? arr[n - 1] : minRest;
}

int main() {
    int arr[] = {3, 1, 4, 1, 5, 9};
    printf("Min: %d\n", findMin(arr, 6));
    return 0;
}
```

Explanation: Compares last element with min of the rest recursively.

---

35. Find power of a number using recursion

c
CopyEdit
```
#include <stdio.h>

int power(int base, int exp) {
    if (exp == 0) return 1;
    return base * power(base, exp - 1);
}

int main() {
    printf("Power: %d\n", power(2, 3));
    return 0;
}
```

Explanation: Multiplies base recursively until exponent reaches zero.

---

36. Convert a number to its sum of digits using recursion

c
CopyEdit
```
#include <stdio.h>

int sumDigits(int n) {
    if (n == 0) return 0;
    return (n % 10) + sumDigits(n / 10);
}

int main() {
    printf("Sum of digits: %d\n", sumDigits(1234));
    return 0;
}
```

Explanation: Adds last digit to sum of remaining digits recursively.

---

37. Reverse a number using recursion

c
CopyEdit
```
#include <stdio.h>

int reverseNumber(int n, int rev) {
    if (n == 0) return rev;
    return reverseNumber(n / 10, rev * 10 + n % 10);
}

int main() {
    printf("Reversed: %d\n", reverseNumber(1234, 0));
    return 0;
}
```

Explanation: Builds reversed number by shifting previous result left and adding last digit.

---

38. Find the nth triangular number using recursion

c
CopyEdit
```
#include <stdio.h>

int triangularNumber(int n) {
    if (n == 1) return 1;
    return n + triangularNumber(n - 1);
}

int main() {
    printf("Triangular Number: %d\n", triangularNumber(5));
    return 0;
}
```

Explanation: Uses formula `T(n) = n + T(n-1)` recursively.

---

39. Count occurrences of a digit in a number

c
CopyEdit
```
#include <stdio.h>

int countDigitOccurrences(int n, int digit) {
    if (n == 0) return 0;
    return ((n % 10 == digit) ? 1 : 0) + countDigitOccurrences(n
/ 10, digit);
}

int main() {
    printf("Occurrences: %d\n", countDigitOccurrences(122333, 
3));
    return 0;
}
```

---

40. Generate Fibonacci sequence up to n terms using recursion

c
CopyEdit

```
#include <stdio.h>

void fibonacci(int n, int a, int b) {
    if (n == 0) return;
    printf("%d ", a);
    fibonacci(n - 1, b, a + b);
}

int main() {
    fibonacci(10, 0, 1);
    return 0;
}
```