Common Recursion Problems (with C++ Implementation)

1. Factorial of a Number

Factorial of n (denoted as `n!`) is the product of all positive integers from 1 to n.

- Formula: `n! = n × (n-1)!`
- Base case: `0! = 1`

```
#include <iostream>
using namespace std;

int factorial(int n) {
    if (n == 0) return 1;   // base case
    return n * factorial(n - 1);
}

int main() {
    cout << "Factorial of 5 = " << factorial(5);
    return 0;
}
```

Output: `Factorial of 5 = 120`

2. Fibonacci Series

The Fibonacci sequence is: `0, 1, 1, 2, 3, 5, 8, ...`

- Formula: `F(n) = F(n-1) + F(n-2)`
- Base cases: `F(0)=0`, `F(1)=1`

```
#include <iostream>
using namespace std;

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

int main() {
    for (int i = 0; i < 8; i++) {
        cout << fibonacci(i) << " ";
    }
    return 0;
}
```

Output: `0 1 1 2 3 5 8 13`

3. Sum of Natural Numbers

To find the sum of first n natural numbers.

- Formula: `Sum(n) = n + Sum(n-1)`

```
#include <iostream>
using namespace std;

int sum(int n) {
    if (n == 0) return 0;   // base case
    return n + sum(n - 1);
}

int main() {
    cout << "Sum of first 10 numbers = " << sum(10);
    return 0;
}
```

Output: `Sum of first 10 numbers = 55`

4. Reverse a String

Swap characters from the start and end, and recursively move towards the center.

```
#include <iostream>
using namespace std;

void reverseString(string &s, int i, int j) {
    if (i >= j) return;
```

```
    swap(s[i], s[j]);
    reverseString(s, i + 1, j - 1);
}

int main() {
    string str = "recursion";
    reverseString(str, 0, str.size() - 1);
    cout << "Reversed string = " << str;
    return 0;
}
```

Output: Reversed string = noisrucer

5. Palindrome Check (String)

A string is palindrome if it reads the same forward and backward.

```
#include <iostream>
using namespace std;

bool isPalindrome(string s, int i, int j) {
    if (i >= j) return true;
    if (s[i] != s[j]) return false;
    return isPalindrome(s, i + 1, j - 1);
}

int main() {
    string str = "madam";
    if (isPalindrome(str, 0, str.size() - 1))
        cout << str << " is a palindrome";
    else
        cout << str << " is not a palindrome";
    return 0;
```

```
}
```

Output: `madam is a palindrome`

6. Sum of Digits

Take the last digit using `% 10`, then call recursion for remaining digits.

```
#include <iostream>
using namespace std;

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

int main() {
    cout << "Sum of digits of 1234 = " << sumOfDigits(1234);
    return 0;
}
```

Output: `Sum of digits of 1234 = 10`

7. Greatest Common Divisor (GCD)

Using **Euclidean Algorithm**:
`GCD(a, b) = GCD(b, a % b)`

```
#include <iostream>
using namespace std;
```

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

int main() {
    cout << "GCD of 36 and 60 = " << gcd(36, 60);
    return 0;
}
```

Output: GCD of 36 and 60 = 12

8. Binary Search

Divide the array in half, check middle element, then search left or right half.

```
#include <iostream>
using namespace std;

int binarySearch(int arr[], int l, int r, int x) {
    if (l > r) return -1;
    int mid = (l + r) / 2;
    if (arr[mid] == x) return mid;
    if (arr[mid] > x) return binarySearch(arr, l, mid - 1, x);
    return binarySearch(arr, mid + 1, r, x);
}

int main() {
    int arr[] = {2, 4, 6, 8, 10, 12};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 10;
    int result = binarySearch(arr, 0, n - 1, target);
```

```
    if (result != -1)
        cout << "Element found at index " << result;
    else
        cout << "Element not found";
    return 0;
}
```

Output: `Element found at index 4`

9. Tower of Hanoi

- Move n disks from source → destination using auxiliary rod.
- Steps:
    1. Move n-1 disks from source → auxiliary.
    2. Move last disk from source → destination.
    3. Move n-1 disks from auxiliary → destination.

```
#include <iostream>
using namespace std;

void towerOfHanoi(int n, char from, char to, char aux) {
    if (n == 0) return;
    towerOfHanoi(n - 1, from, aux, to);
    cout << "Move disk " << n << " from " << from << " to " << to <<
endl;
    towerOfHanoi(n - 1, aux, to, from);
}

int main() {
    int n = 3;
```

```
    towerOfHanoi(n, 'A', 'C', 'B');
    return 0;
}
```

Output (for 3 disks):

```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```