
🧩 Exercise 5: Recursion - Tower of Hanoi

📝 Problem Statement:

Solve the classic **Tower of Hanoi** problem using recursion.

Instructions:

1. Write a method:

```
void TowerOfHanoi(int n, char from, char to, char aux)
```

2. Input:
   - Number of disks `n`
3. Output:
   - Print the steps to move all disks from **source** to **destination** using **auxiliary** rod
4. Example for `n = 3`:

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

🧩 Exercise 6: Greedy Algorithm - Coin Change

📝 Problem Statement:

Use a **greedy algorithm** to find the **minimum number of coins** needed to make a given amount.

Instructions:

1. Available denominations: `{1, 2, 5, 10, 20, 50, 100, 200, 500}`
2. Input:
   - Amount (e.g., `880`)
3. Output:
   - List of coins used and total count
4. Example:

```
Coins used: 500, 200, 100, 50, 20, 10
Total coins: 6
```

🧩 Exercise 7: Hashing - First Non-Repeating Character

🧩 Problem Statement:

Find the **first non-repeating character** in a string using hashing.

Instructions:

1. Input:
    - A string (e.g., `"swiss"`)
2. Output:
    - The first character that appears only once (e.g., `'w'`)
3. Hint:
    - Use a `Dictionary<char, int>` to count frequencies
    - Loop again to find the first char with frequency `1`

🧩 Exercise 8: Backtracking - N-Queens Problem

🧩 Problem Statement:

Solve the **N-Queens problem** using backtracking.

Instructions:

1. Write a method to place `N` queens on an `N x N` chessboard so that no two queens threaten each other.
2. Input:
    - Integer `N` (e.g., `4`)
3. Output:
    - Print all valid board configurations
4. Example for `N = 4`:

```
_ Q _ _
_ _ _ Q
Q _ _ _
_ _ Q _
```
