2nd Task in Embedded Systems

3- Wonderful Number

https://codeforces.com/group/MWSDmqGsZm/contest/223205/submission/312734221

4- Print From 1 To N

https://codeforces.com/group/MWSDmqGsZm/contest/223339/submission/312735911

5- Print Digits using Recursion

https://codeforces.com/group/MWSDmqGsZm/contest/223339/submission/312734944

6- Fibonacci

https://codeforces.com/group/MWSDmqGsZm/contest/223339/submission/312735830

8- 3n + 1 sequence

https://codeforces.com/group/MWSDmqGsZm/contest/223339/submission/312735426

Bonus: Reach Value

https://codeforces.com/group/MWSDmqGsZm/contest/223339/submission/312735974

Summarization

- **C** supports arithmetic operations like addition (+), subtraction (-), multiplication (*), division (/), and modulus (%).

- **Recursion** is an elegant programming paradigm where a function solves a problem by calling itself on smaller instances of the same problem. This approach mirrors mathematical induction and is particularly useful for problems with inherent self-similarity.

- **Fundamental Concepts in Recursion:**

  Base Case: The terminating condition that stops recursion.

  Recursive Case: The step where the function calls itself with a modified argument.

- **Advantages of Recursion:**

1. **Simple Code** – Makes solving complex problems easier.
2. **Shorter Code** – Needs fewer lines than loops.
3. **Good for Some Problems** – Works well for trees, sorting, and backtracking.
4. **Divide & Conquer –** Breaks big problems into smaller ones.

- **Disadvantages of Recursion:**

1. **Uses More Memory** – Every function call takes extra space.
2. **Slower** – Calling functions repeatedly takes time.
3. Hard to Debug – Tracking the function flow is tricky.
4. **Can Crash** – If there's no proper stopping point (base case), it keeps running forever.
5. **Loops Can Be Better** – Sometimes, a loop is faster and uses less memory.

- **When to Use Recursion?**

    1. The problem can be broken into smaller subproblems of the same type.
    2. The problem requires backtracking.

3. The recursive solution is more readable and maintainable.

- **Types of Recursion in C:**
    1. **Direct Recursion –** The function calls itself directly within its body.
    2. **Indirect Recursion –** A function does not call itself directly but calls another function, which then calls the original function, creating a cycle.
    3. **Tail Recursion** – The recursive call is the last operation performed before returning the result, making it efficient in memory usage.
    4. **Head Recursion** – The function makes the recursive call first, and only after returning does it perform other operations.
    5. **Tree Recursion** – A function makes multiple recursive calls, leading to a branching structure similar to a tree.
    6. **Nested Recursion** – The function's recursive call contains another recursive call as its argument, leading to deep recursion levels.