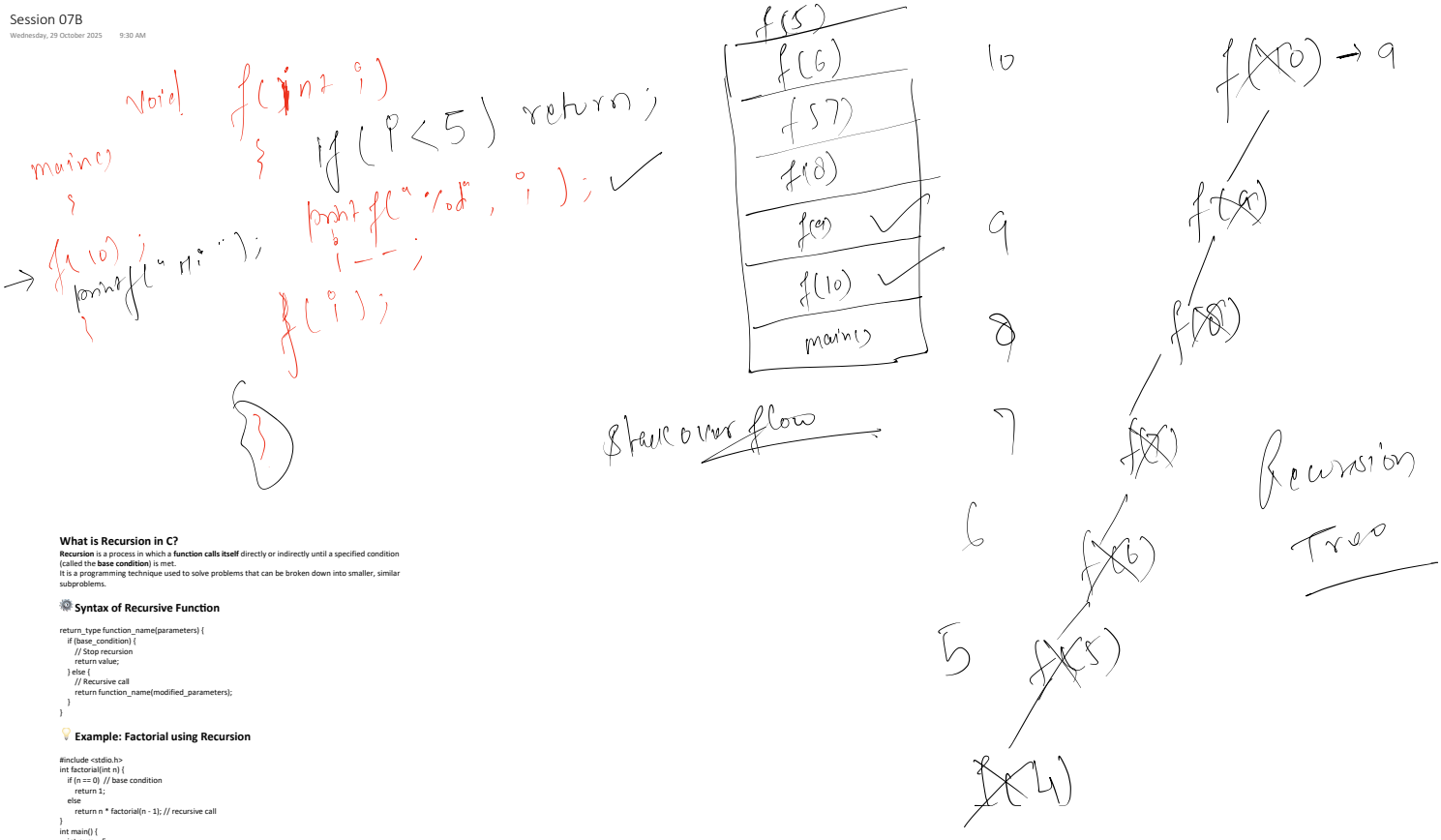
Session 07B

Wednesday, 29 October 2025 9:30 AM



What is Recursion in C?

Recursion is a process in which a **function calls itself** directly or indirectly until a specified condition (called the **base condition**) is met.
It is a programming technique used to solve problems that can be broken down into smaller, similar subproblems.

⚙️ Syntax of Recursive Function

```
return_type function_name(parameters) {
    if (base_condition) {
        // Stop recursion
        return value;
    } else {
        // Recursive call
        return function_name(modified_parameters);
    }
}
```

💡 Example: Factorial using Recursion

```
#include <stdio.h>
int factorial(int n) {
    if (n == 0)  // base condition
        return 1;
    else
        return n * factorial(n - 1); // recursive call
}
int main() {
    int num = 5;
    printf("Factorial of %d is %d", num, factorial(num));
    return 0;
}
```

Output:

```
Factorial of 5 is 120
```

🌟 Advantages of Recursion

1. **Simplifies Code:**
   Makes code shorter and easier to understand for problems that have repetitive patterns.
2. **Solves Complex Problems Easily:**
   Ideal for problems like **tree traversal, factorial, Fibonacci, and divide-and-conquer algorithms**.
3. **Reduces Code Size:**
   Removes the need for complex loops and extra variables.
4. **Mathematically Elegant:**
   Directly represents mathematical formulas (e.g., factorial, Fibonacci, etc.).

⚠️ Disadvantages of Recursion

1. **Higher Memory Usage:**
   Each recursive call adds a new function call to the **call stack**, consuming more memory.
2. **Slower Execution:**
   Due to multiple function calls and stack operations.
3. **Risk of Stack Overflow:**
   If base condition is missing or incorrect, recursion can go into **infinite calls** and crash the program.
4. **Difficult Debugging:**
   Tracing recursive calls can be complex compared to iterative solutions.

🧠 Uses of Recursion

Recursion is useful in:

1. **Mathematical computations:** Factorial, Fibonacci series, Power functions.
2. **Data structures:** Traversing **trees**, **graphs**, and **linked lists**.
3. **Divide and Conquer algorithms:**
   - Merge Sort
   - Quick Sort
   - Binary Search
4. **Backtracking problems:**
   - Tower of Hanoi
   - N-Queens problem
   - Maze Solving

```
void printRange(int start, int end)
{
    if (start > end)
        return;
    printf("%d\n", start);
    printRange(start + 1, end);
}
```