Functions in DSA

A **function** is a self-contained block of code designed to perform a specific task. Functions help break down complex problems into manageable chunks, improving code modularity and reusability. In DSA, functions are crucial for implementing algorithms and dividing tasks efficiently.

Key Concepts:

1. **Function Definition:** The part where the function's name, parameters, and return type are declared.
    - Syntax:

```
return_type function_name(parameters) {
    // code
}
```

2. **Function Declaration (or Prototype):** Informs the compiler about the function before its actual definition.
3. **Function Call:** Invokes the function to perform its task.
4. **Parameter Passing:**
    - **Pass by Value:** A copy of the argument is passed.
    - **Pass by Reference:** The actual memory address is passed, allowing modifications.
5. **Return Type:** Determines what data type the function will return. If no value is returned, use void.
6. **Recursion:** A function that calls itself. Used to solve problems like factorial, Fibonacci, and divide-and-conquer algorithms.

Important Questions for Practice:

1. Write a function to calculate the factorial of a number.
2. Explain the difference between pass by value and pass by reference.
3. Implement a recursive function to find the nth Fibonacci number.
4. Write a function to reverse a string.
5. What are the advantages of using functions in programming?