**Recursion**

**Key Methods in Recursion**

* ·  **IBH Method:**  
   - Induction Step  
   - Base Case  
   - Hypothesis

**· Input/Output Approach**

**· Recursive Tree Approach**

# **Problem 1: Print Numbers from 1 to N**

### Objective: Print all numbers from 1 to `n` in ascending order using recursion.

IBH Design:

**void printAscending(int n) {  
 // Base Case  
 if (n == 1) {  
 cout << 1 << " ";  
 return;  
 }  
  
 // Hypothesis (Assume printAscending(n-1) works correctly)  
 printAscending(n - 1);  
  
 // Induction Step  
 cout << n << " ";  
 }**

Example:  
 **Input: n = 10  
 Output: 1 2 3 4 5 6 7 8 9 10**

# **Problem 2: Print Numbers from N to 1**

### IBH Design:

**void printDescending(int n) {  
 // Base Case  
 if (n == 1) {  
 cout << 1 << " ";  
 return;  
 }  
  
 // Induction Step  
 cout << n << " ";  
   
 // Hypothesis  
 printDescending(n - 1);  
 }**

Example:  
 Input: n = 10  
 Output: 10 9 8 7 6 5 4 3 2 1

# **Problem 3: Factorial of N**

IBH Design:

**int factorial(int n) {  
 // Base Case  
 if (n <= 1) {  
 return 1;  
 }  
  
 // Hypothesis + Induction  
 return n \* factorial(n - 1);  
 }**

Example:  
 Input: n = 5  
 Output: 120

# **Problem 4: Height of a Binary Tree**

## Definition: Height of a tree = longest path from root to leaf.

### IBH Design:

**struct Node {  
 int data;  
 Node\* left;  
 Node\* right;  
 };  
  
 int height(Node\* root) {  
 // Base Case  
 if (root == nullptr) {  
 return 0;  
 }  
  
 // Hypothesis  
 int leftHeight = height(root->left);  
 int rightHeight = height(root->right);  
  
 // Induction  
 return 1 + max(leftHeight, rightHeight);  
 }**

# **Problem 5: Sort an Array Using Recursion**

Given:  
 vector<int> arr = {1, 0, 5, 2};

IBH Design:

**void insert(vector<int>& v, int temp) {  
 // Base Case  
 if (v.size() == 0 || v.back() <= temp) {  
 v.push\_back(temp);  
 return;  
 }  
  
 // Hypothesis  
 int last = v.back();  
 v.pop\_back();  
  
 insert(v, temp);  
  
 // Induction Step  
 v.push\_back(last);  
 }  
  
 void sortRecursive(vector<int>& v) {  
 // Base Case  
 if (v.size() <= 1) {  
 return;  
 }  
  
 // Hypothesis  
 int temp = v.back();  
 v.pop\_back();  
  
 sortRecursive(v);  
  
 // Induction Step  
 insert(v, temp);  
 }**

**➤ Main Function:**

**int main() {  
 vector<int> arr = {1, 0, 5, 2};  
  
 sortRecursive(arr);  
  
 for (int i : arr) {  
 cout << i << " ";  
 }  
  
 return 0;  
 }**

**Output: 0 1 2 5**

# **Summary of Concepts**

| Concept | Description |
| --- | --- |
| Base Case | Simplest version of the problem (stop condition) |
| Hypothesis | Assume the recursion works for a smaller input |
| Induction Step | Use the assumption to solve the current problem |
| Recursive Tree | Visualizes the breakdown of function calls |