Chapter II : The Lists- Linked Lists

Application on C Language, Simple Traversal Functions

Definition of a List in C

In **C programming**, a **list** is a data structure used to store multiple elements dynamically. Since C does not have a built-in list type like Python or Java, lists are typically implemented using **arrays** or **linked lists**.

The Linked Lists

```c
#include <stdio.h>
#include <stdlib.h>

// Define structure for a node
struct Node {
    int data;
    struct Node* next;
};

// Function to print the list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
```

1. Simple Traversal Functions

Traversal means visiting each node in the linked list and performing an operation, such as printing the values.

Example: Traversing and Printing a Linked List

```c
#include <stdio.h>
#include <stdlib.h>

// Definition of a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;
```

Function to traverse and print the list

```c
// Function to traverse and print the list
void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

Function to add a node at the beginning of the list

```c
// Function to add a node at the beginning of the list
Node* addAtHead(Node* head, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = head;
    return newNode;
}
```

Main program

```c
// Main program
int main() {
    Node* head = NULL;

    // Adding some elements
    head = addAtHead(head, 3);
    head = addAtHead(head, 2);
    head = addAtHead(head, 1);

    // Printing the list
    printList(head);

    return 0;
}
```

Output:

```
1 -> 2 -> 3 -> NULL
```