Khushi m panjwani/D10B-42

AIM- To understand the concept of singly linked list.

THEORY- A **singly linked list** is a data structure where each element (node) contains data and a pointer to the next node in the sequence.

Structure

- **Node**:
  - **Data**: Stores the value.
  - **Next**: Points to the next node.
- **Head**: Pointer to the first node of the list. If the list is empty, the head is `NULL`.

Operations

1. **Insertion**:
   - **At the Beginning**: Create a new node, set its `next` to the current head, and update head.
   - **At the End**: Traverse to the last node and link the new node.
   - **At a Specific Position**: Traverse to the desired position, update pointers to insert the new node.
2. **Deletion**:
   - **From the Beginning**: Update head to the next node and free the old head.
   - **From the End**: Traverse to the second-to-last node, update its `next` to `NULL`, and free the last node.
   - **From a Specific Position**: Update pointers to bypass the node to be deleted and free it.
3. **Traversal**:
   - Start from head and follow `next` pointers to visit all nodes.
4. **Search**:
   - Traverse from head, comparing each node's data to find a target value.

Advantages

- **Dynamic Size**: Can easily grow or shrink.
- **Efficient Insertion/Deletion**: Simple if position is known.

Disadvantages

- **Memory Overhead**: Extra memory needed for pointers.
- **Sequential Access**: Accessing elements takes O(n) time.

Use Cases

- Useful when the number of elements changes frequently or when insertions and deletions are more common than direct access.

Khushi m panjwani/D10B-42

INPUT-

```
#include <stdio.h>

#include <stdlib.h>


// Define the structure for a node in the linked list

typedef struct Node {

    int data;

    struct Node* next;

} Node;


// Function to create a new node with given data

Node* createNode(int data) {

    Node* newNode = (Node*)malloc(sizeof(Node)); // Allocate memory for the node

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        exit(1);

    }

    newNode->data = data; // Set the data for the node

    newNode->next = NULL; // Initialize the next pointer to NULL

    return newNode;

}


// Function to append a new node to the end of the list
```

Khushi m panjwani/D10B-42

```
void append(Node** head, int data) {
    Node* newNode = createNode(data); // Create a new node
    if (*head == NULL) {
        *head = newNode; // If the list is empty, set the new node as the head
        return;
    }
    Node* current = *head;
    while (current->next != NULL) {
        current = current->next; // Traverse to the end of the list
    }
    current->next = newNode; // Link the new node to the end of the list
}


// Function to print all elements of the list
void display(Node* head) {
    Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data); // Print the current node's data
        current = current->next; // Move to the next node
    }
    printf("NULL\n"); // End of the list
}


// Main function
```

Khushi m panjwani/D10B-42

```
int main() {
    Node* head = NULL; // Initialize the head of the list to NULL
    int n, data;

    // Prompt user for the number of elements
    printf("Enter the number of elements to add to the list: ");
    scanf("%d", &n);

    // Read and append each element to the list
    for (int i = 0; i < n; i++) {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &data);
        append(&head, data);
    }

    // Display the list
    printf("List: ");
    display(head);

    // Free the allocated memory
    Node* current = head;
    Node* nextNode;
    while (current != NULL) {
        nextNode = current->next; // Store the next node
```

```
        free(current); // Free the current node
        current = nextNode; // Move to the next node
    }

    return 0;
}
```

OUTPUT-

```
Enter the number of elements to add to the list: 3
Enter element 1: 5
Enter element 2: 79
Enter element 3: 1
List: 5 -> 79 -> 1 -> NULL
```

CONCLUSION-

In this way we understand the concept of singly linked list in easy and understandable way.