Experiment No-02: Introduction to Linked List.

Objectives

- Introduce with the single linked list.
- Create a singly linked list.
- Learn insertion operation in the singly linked list.

Prerequisite: [Function], [Pointer], and [Structure].

Example 1: Create a Single Linked List.

```
#include<iostream>
#include<stdlib.h>
#include<bits/stdc++.h>

using namespace std;

// Create a Node Data Type
struct Node{
    int data;
    Node *next;
};


int main()
{
    //Initialize three nodes with NULL pointer
    Node *a =NULL,*b=NULL,*c=NULL;

    // Allocate Memory for each node
    a = (Node*) malloc(sizeof(Node));
    b = (Node*) malloc(sizeof(Node));
    c = (Node*) malloc(sizeof(Node));

    // Insert Data and Connect the nodes
    a->data = 10;
    b->data = 20;
    c->data = 30;
    a->next = b;
    b->next = c;
    c->next = NULL;

    //Traverse the Linked list
    while (a!=NULL){
        cout<<a->data<<" ";
        a = a->next;
    }
}
```

Example 2: Create a linked list from an array and return the head.

```
#include<bits/stdc++.h>
using namespace std;

// Create a Node Data Type
struct Node
{
    int data;
    Node *next;

    // Set Node value and next pointer
    Node(int x)
    {
        data = x;
        next = nullptr;
    }
};
// Create Linked List Function
Node* constructLL(int arr[], int arrsize) {

    Node *head = new Node(arr[0]); // new work as a malloc function
    Node *current = head; // keep track of the new node

    for (int i = 1; i<arrsize; i++)
    {
        Node *temp = new Node(arr[i]); // new node
        current->next = temp;
        current = temp;
    }
    return head;
}
// Traverse Function
void TraverseList(Node *head)
{
    while (head!=nullptr)
    {
        cout<<head->data<<" ";
        head = head->next;
    }
}
// Main Function
int main()
{
    int arr [8] = {2,4,5,6};
    // Construct Linked List
    Node *head = constructLL(arr,4);
    // Print the List
    TraverseList(head);
}
```

Example 3: Insert a node at the beginning of the list.

```
#include<bits/stdc++.h>
using namespace std;
// This program only included the Function

Node* insertAtFirst(Node* head, int newValue) {

    Node* current = nullptr;
    // Edge Case: The list could be empty
    if (head==nullptr)
    {
        current = new Node(newValue);
        head = current;

        return head;
    }

    current = new Node(newValue);
    current->next = head;
    head = current;

    return head;

}
```

Example 3: Insert a node at the end of the list. [Assume the list already has two nodes.]

```
#include<bits/stdc++.h>
using namespace std;
// This program only included the Function

/* Possible Edge Cases:
1) The list could be empty
2) The list has only one node
3) The list has more than one node
*/
Node* insertAtLast(Node* head, int newValue) {
    Node *temp = head, *current;

    while (temp->next!= nullptr)
    {
           temp = temp->next;
    }
    current = new Node(newValue);
    temp->next = current;

    return head;

}
```

Practice Exercise

1. Write a C++ program to insert a new node at the end of a Singly Linked List [Consider all edge cases].
2. Write a C++ program to find the length of a singly linked list.
3. Write a C++ program to delete the first node of a Singly Linked List.
4. Write a C++ program to delete the last node of a Singly Linked List.

Resources (Link)

Please try to solve similar problems at an online Judge.

1. Create Linked List
2. Insert a Node
3. Delete a Node