Week #3: In-Class Exercises on Linked List

1. **Create a linked list** Create a singly Linked List with the following elements: 10->20->30->40

```
 Node* head = new Node;
 head->data = 10;
 head->next = NULL;

 head->next = new Node;
 head->next->data = 20;
 head->next->next = NULL;

 head->next->next = new Node;
 head->next->next->data = 30;
 head->next->next->next = NULL;

 head->next->next->next = new Node;
 head->next->next->next->data = 40;
 head->next->next->next->next = NULL;
```

2. **Display Linked List** Write a function to display the elements of a linked list.

```
void display(Node* p){
    Node* current = p;
    while(current != NULL){
        cout<<current->data<<endl;
        current = current->next;
    }
}
```

3. **Count Nodes** Write a function to count and return the number of nodes in the linked list.

```
int count(Node* head)
{
    Node* current = head;
    int i = 0;
    while(current != NULL){
        i++;
        current = current->next;
    }

    return i;
}
```

4. **Find an Element** Write a function to find whether a specific element (e.g., 20) exists in the linked list.

```
bool find(Node* head, int item)
{
    Node* current = head;
    while(current != NULL)
    {
        if(current->data == item)
            return true;
        current = current->next;
    }
    return false;
}
```

5. **Insert at the Beginning** Write a function to insert the given node at the beginning of the linked list.

```
void insertFirst(Node*& head, int item)
{
    Node* newNode = new Node;
    newNode->data = item;
    newNode->next = head;
    head = newNode;
}
```

6. **Insert at the End** Write a function to insert the given node at the end of the linked list.

```
void insertLast(Node*& head, int item)
{
    Node* newNode = new Node;
    newNode->data = item;
    newNode->next = NULL;

    if(head == NULL)
    {
        head = newNode;
    }else{
        Node* current = head;
        while(current->next != NULL)
        {
            current = current->next;
        }
        current->next = newNode;
    }
}
```

7. **Delete a Node** Write a function to delete the node from the linked list.

```
void deleteItem(Node*& head, int item)
{
    if(head == NULL)
        return;
    if(head->data == item)
    {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Node* current = head;
    while(current->next != NULL && current->next->data != item)
    {
        current = current->next;
    }
    if(current->next == NULL)
        return;
    Node* temp = current->next;
    current->next = current->next->next;
    delete temp;
}
```

8. **Destroy the list** Write a function to deallocate all linked list nodes

```
void destroy(Node*& head)
{
     while(head != NULL)
     {
        Node* temp = head;
        head = head->next;
        delete temp;
     }
}
```

You can find the full program below.

```cpp
#include <iostream>

using namespace std;

struct Node
{
    int data;
    Node* next;
};

void display(Node*);
int count(Node*);
bool find(Node*, int);
void insertFirst(Node*&, int);
void insertLast(Node*&, int);
void deleteItem(Node*&, int);
void destroy(Node*&);

int main()
{
    Node* head = new Node;
    head->data = 10;
    head->next = NULL;

    head->next = new Node;
    head->next->data = 20;
    head->next->next = NULL;

    head->next->next = new Node;
    head->next->next->data = 30;
    head->next->next->next = NULL;

    head->next->next->next = new Node;
    head->next->next->next->data = 40;
    head->next->next->next->next = NULL;

    count(head);
    display(head);
    cout<<"Count:"<<count(head)<<endl;
    bool isFound = find(head, 10);
    cout<<"Searc for 10:"<<isFound<<endl;//expected 1

    isFound = find(head, 80);
    cout<<"Searc for 80:"<<isFound<<endl;//expected 0

    insertFirst(head, 5);
    display(head);
    cout<<"Count:"<<count(head)<<endl;

    insertLast(head, 100);
    display(head);
    cout<<"Count:"<<count(head)<<endl;

    deleteItem(head, 40);
    display(head);

    destroy(head);
}

void display(Node* p){
    Node* current = p;
    while(current != NULL){
        cout<<current->data<<endl;
        current = current->next;
    }
}

int count(Node* head)
{
    Node* current = head;
    int i = 0;
    while(current != NULL){
        i++;
        current = current->next;
    }

    return i;
}


bool find(Node* head, int item)
{
    Node* current = head;
    while(current != NULL)
    {
        if(current->data == item)
            return true;
        current = current->next;
    }
    return false;
}

void insertFirst(Node*& head, int item)
{
    Node* newNode = new Node;
    newNode->data = item;
    newNode->next = head;
    head = newNode;
}
```

```
void deleteItem(Node*& head, int item)
{
    if(head == NULL)
        return;
    if(head->data == item)
    {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Node* current = head;
    while(current->next != NULL && current->next->data != item)
    {
        current = current->next;
    }
    if(current->next == NULL)
        return;
    Node* temp = current->next;
    current->next = current->next->next;
    delete temp;
}

void destroy(Node*& head)
{
     while(head != NULL)
     {
        Node* temp = head;
        head = head->next;
        delete temp;
     }
}
```