```
#include<bits/stdc++.h>
using namespace std;
struct Node
{
    int data;
    struct Node *next;
};
struct Node *head=NULL;

void firstNode(int data)
{
    head=(struct Node*)malloc(sizeof(struct Node));
    head->data=data;
    head->next=NULL;
}

void addNode(int data)
{
    struct Node *ptr;
    ptr=head;
    struct Node *temp=NULL;
    temp=(struct Node*)malloc(sizeof(struct Node));
    temp->data=data;
    temp->next=NULL;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    ptr->next=temp;
}

void printNode()
{
    struct Node *temp;
    temp=head;
    cout<<"Linked List"<<endl;
    while(temp!=NULL)
    {
        cout << temp->data << "->";
        temp=temp->next;
    }
}

int main()
{
    firstNode(5);
    addNode(10);
    addNode(15);
    printNode();

    return 0;
}
```

```
#include<bits/stdc++.h>
using namespace std;
struct Node
{
    int data;
    struct Node *next;
};
struct Node *head=NULL;


void firstNode(int data)
{
    head=(struct Node*)malloc(sizeof(struct Node));
    head->data=data;
    head->next=NULL;
}

void addNode(int data)
{
    struct Node *ptr;
    ptr=head;
    struct Node *temp=NULL;
    temp=(struct Node*)malloc(sizeof(struct Node));
    temp->data=data;
    temp->next=NULL;
    while(ptr->next!=NULL)
    {
        ptr=ptr->next;
    }
    ptr->next=temp;
}


void add_Node_n_position(int position,int data)
{
    struct Node *ptr=NULL;
    struct Node *temp=NULL;
    ptr=head;
    temp=(struct Node*)malloc(sizeof(struct Node));
    temp->data=data;
    temp->next=NULL;
    int i=0;
    while(i!=position-1)
    {
        ptr=ptr->next;
        i++;
    }
    temp->next=ptr->next;
    ptr->next=temp;

}


void del_n_position(int position)
{
    if(head->next==NULL)
    {
        head=NULL;
    }
```

```
    else{
        struct Node *ptr;
        ptr=head;
        int i=0;
        while(i!=position-1)
        {
            ptr=ptr->next;
            i++;
        }
        ptr->next=ptr->next->next;
    }
}


void printNode()
{
    struct Node *temp;
    temp=head;
    printf("Linked List: ");
    while(temp!=NULL)
    {
        cout<<temp->data <<"->";
        temp=temp->next;
    }
}

int main()
{
    firstNode(5);
    addNode(10);
    addNode(15);
    printNode();
    cout <<"\n \n" << endl;
    printf("After Extend Link: \n");
    add_Node_n_position(2,25);
    del_n_position(1);
   add_Node_n_position(2,20);
    del_n_position(3);
    printNode();
    cout << "\n\n\n" << endl;

    return 0;
}
```

```
#include<bits/stdc++.h>
using namespace std;
struct Node{
    int data;
    struct Node *link;
};
struct Node *head=NULL;

void firstNode(int data)
{
    head=(struct Node*)malloc(sizeof(struct Node));
    head->data=data;
    head->link=NULL;
}
void addNode(int data)
{
    struct Node*temp;
    struct Node *ptr;
    ptr=head;
    temp=(struct Node*)malloc(sizeof(struct Node));
    temp->data=data;
    temp->link=NULL;
    while(ptr->link!=NULL)
    {
        ptr=ptr->link;
    }
    ptr->link=temp;
}

void reverse()
{
    struct Node *prev=NULL;
    struct Node *ptr=head;
    struct Node *next=NULL;
    while(ptr!=NULL)
    {
        next=ptr->link;
        ptr->link=prev;
        prev=ptr;
        ptr=next;
    }
    head=prev;

}

void printNode()
{
    struct Node *ptr;
    ptr=head;
    while(ptr!=NULL)
    {
        cout << ptr->data <<"->";
        ptr=ptr->link;
    }

}

int main()
{
```

```
    cout << "Linked List: "<< endl;
    firstNode(5);
    addNode(25);
    addNode(20);
    printNode();
     cout << "\n\n"<<"->";
    printf("Reverse the linked list: ");
    reverse();
    printNode();
    cout << "\n\n" ;

    return 0;

}
```

```
#include <bits/stdc++.h>
using namespace std;
struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(nullptr) {}
};

class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next)
            return false;

        ListNode *slow = head;
        ListNode *fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
                return true;
        }

        return false;
    }
};


ListNode* createLinkedListWithCycle() {
    ListNode* head = new ListNode(1);
    ListNode* node2 = new ListNode(2);
    ListNode* node3 = new ListNode(3);
    ListNode* node4 = new ListNode(4);
    ListNode* node5 = new ListNode(5);
    ListNode* node6 = new ListNode(6);
    ListNode* node7 = new ListNode(7);
    ListNode* node8 = new ListNode(8);
    ListNode* node9 = new ListNode(9);
    ListNode* node10 = new ListNode(10);

    head->next = node2;
    node2->next = node3;
    node3->next = node4;
    node4->next = node5;
    node5->next = node6;
    node6->next = node7;
    node7->next = node8;
    node8->next = node9;
    node9->next = node10;

    node10->next = node4;

    return head;
}

int main() {
    Solution sol;
```

```
    ListNode* head = createLinkedListWithCycle();

    bool hasCycle = sol.hasCycle(head);
    cout << "Has Cycle: " << (hasCycle ? "Yes" : "No") << endl;


    if (hasCycle) {
        ListNode *slow = head;
        ListNode *fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
                break;
        }

        slow = head;
        while (slow != fast) {
            slow = slow->next;
            fast = fast->next;
        }

        cout << "Cycle Start Node: " << slow->val << endl;
    }

    return 0;
}
```

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


struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(nullptr) {}
};

class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode temp(0);
        ListNode* tail = &temp;

        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next;
        }

        tail->next = l1 ? l1 : l2;

        return temp.next;
    }
};


void printList(ListNode* head) {
    while (head) {
        cout << head->val;
        if (head->next) cout << " -> ";
        head = head->next;
    }
      cout << endl;
}

int main() {
    Solution sol;


    ListNode* listA = new ListNode(5);
    listA->next = new ListNode(10);


    ListNode* listB = new ListNode(7);
    listB->next = new ListNode(12);

    cout << "List A: ";
    printList(listA);

    cout << "List B: ";
    printList(listB);
```

```
    ListNode* mergedList = sol.mergeTwoLists(listA, listB);

    cout << "Merged List: ";
    printList(mergedList);

    return 0;
}
```