# 20220915 数据结构与算法 解题报告

## Delete Duplicate

可将该链表视作多个相同元素链拼接在一起形成的，只需保留每个子链的链头即可。

#include "LinkNode.h"  
#include <bits/stdc++.h>  
using namespace std;  
  
void delete\_duplicate(LinkList &head) {  
 for (LinkList cur = head; cur != NULL; cur = cur->next) {  
 for (LinkList tmp = cur->next; tmp != NULL && cur->data == tmp->data; tmp = cur->next) {  
 cur->next = tmp->next; delete tmp;  
 }  
 }  
}

## Insert for single link list with head node

如题，只需找到指定节点并进行插入操作即可。

#include <bits/stdc++.h>  
#include "ListNode.h"  
using namespace std;  
  
void List::insert(int toadd, int pos) {  
 auto cur = head;  
 for (; --pos; cur = cur->next);  
 auto tmp = new ListNode; tmp->data = toadd; tmp->next = cur->next;  
 cur->next = tmp;  
}

## Remove for single link list with head node

如题，只需找到指定节点并进行删除操作即可。

#include <bits/stdc++.h>  
#include "ListNode.h"  
using namespace std;  
  
void List::remove(int pos) {  
 auto cur = head;  
 for (; --pos; cur = cur->next);  
 auto tmp = cur->next; cur->next = tmp->next; delete tmp;  
}

## Loops in the Linked List

从链头开始遍历，对每个节点打上标记，若遇到已被标记节点则认为该链表含有环，若遍历到尾节点则认为不含有环。

标记节点可利用STL库中unordered\_map实现，以节点地址为下标。

#include <bits/stdc++.h>  
#include "Node.h"  
using namespace std;  
  
bool check(node \*head) {  
 unordered\_map<node\*, int> sgn;  
 for (auto cur = head; cur != NULL; cur = cur->next)   
 if (sgn[cur])  
 return true;  
 else  
 sgn[cur] = true;  
 return false;  
}

## 实验总结

熟练了链表的基本结构及基本操作，能够较好掌握链表这一数据结构及较好理解其使用场景。