11月16日题目讲解

1. 定义一个由数结点构成的单向链表的结点类型，简要说明如何利用该结点类型表示一个单向链表;其次，定义并实现一个函数以一个由整数结点构成的单向链表L为参数，返回一个新的单向链表，新的单向链表由L中删除所有绝对值为素数的结点后剩余结点构成并且，各结点中的整数值从链头至链尾按不增顺序排列

> 思路：
>
> 考察知识点：
>
> 1. 是否会判断素数
> 2. 是否建立链表
> 3. 是否会对列表进行排序
> 4. 是否会删除节点
>
> 四个知识点全部合在一起就是这个题目，这四个函数分别就是本此的得分点。

答案：

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// 定义单向链表的节点类型
typedef struct LNode {
    int data;           // 存储结点数据
    struct LNode *next; // 指向下一个结点的指针
} LNode, *LinkList;

// 判断一个数是否为素数
int IsPrime(int num) {
    if (num <= 1) return 0;
    if (num % 2 == 0 && num > 2) return 0;
    for (int i = 3; i <= sqrt(num); i ++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

// 创建带头结点的单链表
LinkList CreateList(int arr[], int n) {
    LinkList L = (LinkList)malloc(sizeof(LNode));
    L->next = NULL;
    for (int i = 0; i < n; i++) { // 尾插法建立单链表
        LNode *s = (LNode *)malloc(sizeof(LNode));
```

```
        s->data = arr[i];
        s->next = L->next;
        L->next = s;
    }
    return L;
}

// 删除绝对值为素数的结点
void DeletePrimeNodes(LinkList L) {
    LNode *p = L->next, *pre = L;
    while (p) {
        if (IsPrime(abs(p->data))) { // 如果绝对值为素数
            pre->next = p->next; // 删除p结点
            free(p);
            p = pre->next;
        } else { // 不删除
            pre = p;
            p = p->next;
        }
    }
    SortList(L); // 对链表进行排序
}

// 插入排序
void SortList(LinkList L) {
    LNode *L1 = L->next; // L1指向第一个结点
    L->next = NULL; // 初始化排序好的链表为空
    while (L1) {
        LNode *next = L1->next; // 保存下一个结点的地址
        LNode *p = *L;
        // 找到插入位置
        while (p->next && p->next->data < L1->data) {
            p = p->next;
        }
        L1->next = p->next; // 插入L1结点
        p->next = L1;
        L1 = next; // 处理下一个结点
    }
}

// 打印链表
void PrintList(LinkList L) {
    LNode *p = L->next; // 第一个结点
    while (p) {
        printf("%d ", p->data);
        p = p->next;
    }
    printf("\n");
}

// 主函数
int main() {
    int arr[] = {10, 7, 5, 3, 11, 13, 4, 6, 8, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    LinkList L = CreateList(arr, n);
    printf("Original List: ");
```

```
    PrintList(L);

    DeletePrimeNodes(L);
    printf("New List: ");
    PrintList(L);

    return 0;
}
```

2. 定义单链表(每个结点包含2个字段:整数信息、后继指针)，实现函数，删除该单链表中所含整数信息等于整数x的多个重复结点。例如:若单链表中存储的整数信息依次为1、5、5、0、5、6、0、0、5、1，若x为5，则得到的单链表种相应信息依次为1、0、6、0、0、1。

> 思路:
>
> 遍历链表，删除值为x的节点即可

答案:

```
#include <stdio.h>

// 定义单链表的节点类型
typedef struct LNode {
    int data;           // 存储整数信息
    struct LNode *next; // 指向下一个结点的指针
} LNode, *LinkList;

// 删除值为x的重复结点,假设L带头结点
void DeleteDuplicates(LinkList L, int x) {
    LNode *p = L->next, *q = NULL;
    while (p) {
        if (p->data == x) {
            q->next = p->next; // 删除p结点
            free(p);
            p = q->next;
        } else {            //q永远指向p的前一个结点
            q = p;
            p = p->next;
        }
    }
}
```

3. 编写函数删除链表中重复的节点。在一个已经拍序的整数链表中，删除该链表中节点值相同的重复节点，重复节点都不保留，返回处理后的链表。例如:通过参数给定链表:1->2->3->3->4->4->5返回链表:1->2->5

> 思路:
>
> 遍历链表，找到下一个元素不相同的位置，然后当前指针指向下一个不同的即可。

答案:

```
#include <stdio.h>
// 定义单链表的节点类型
typedef struct LNode {
    int data;           // 存储整数信息
    struct LNode *next; // 指向下一个结点的指针
} LNode, *LinkList;

// 删除链表中重复的节点,默认链表带头结点
LinkList DeleteDuplicates(LinkList L) {
    // 辅助指针，始终指向当前结点的前一个结点
    LNode *pre = L;
    // 当前结点
    LNode *current = L->next;

    while (current != NULL) {
        // 查找当前结点值第一次出现的下一个结点
        LNode *p = current->next;
        while (p != NULL && current->data == p->data) {
            p = p->next;
        }
        // 删除当前结点
        pre->next = p;
        pre = p;
        // 移动到下一个结点
        current = p;
    }
    return L;
}
```