

吴老师，骆学长和孙学长好，以下是我的作业。

## 上机随堂作业

### - 教材

P.186 题 7: 合并有序链表 struct Node\* mergeSortedLists(struct Node\* list1, struct Node\* list2)。要求：不允许创建新节点，必须使用原有节点；分别打印两个子链表与合并后的链表。测试样例：list1 = [1,3,5], list2 = [2,4,6]

### - 上机指导书

实验七 5 思考题：删除链表的最值表元。要求：从键盘输入表元，在无序链表找到最值并删除；分别打印原链表与处理后的链表。测试样例：list = [2, 4, 6, 1, 3, 5]

-----  
- 附加题(该题不计入现有分数体系，不要求所有同学完成，学有余力、进阶提升的同学可以进行尝试完成。特别说明：在最终成绩出现平分时，优先考虑该类题目完成的数量和质量较好的同学)

拆分链表：给定一个单链表，链表中的每个节点包含一个整数值，按照以下要求对链表进行拆分和重组：

1. 将所有小于给定值 x 的节点放到链表前面
2. 将所有大于或等于给定值 x 的节点放到链表后面
3. 保持节点的相对顺序不变
4. 必须在原链表的基础上进行操作，不能创建新节点

测试样例：输入：链表节点数 n=6，链表中的值 1, 4, 3, 2, 5, 2，分割值 x=3；输出：1, 2, 2, 4, 3, 5。

输入：链表节点数 n=5，链表中的值 2, 1, 2, 2, 1，分割值 x=2；输出：1, 1, 2, 2, 2。

[Chi-Shan0707/Homework-in-CS10004-Programming-by-yhchi](#)

代码仓库↑

1.

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

struct Node
{
    int val;
    struct Node* next;
};
```

```
// 打印链表
typedef struct Node Node;
void printList(Node* head)
{
    Node* p = head;
    printf("*****\n");
    while(p!=NULL)
    {
```

```

    printf("%d ", p->val);
    p = p->next;
}
printf("\n");
}

Node* mergeSortedLists(Node* head1, Node* head2)
{
    Node *head;
    Node *tail;
    if(head1==NULL) return head2;
    if(head2==NULL) return head1;
    //竟然莫名的像fhq_treap
    if(head1->val<head2->val)
    {

        head1->next=mergeSortedLists(head1->next, head2);
        return head1;
    }
    else
    {
        head2->next=mergeSortedLists(head1, head2->next);
        return head2;
    }
    //这里也像fhq_treap 的merge
}

```

}

```

Node* array_to_list(int* arr, int n)
{
    int i;
    Node* head = (Node*)malloc(sizeof(Node));
    Node* tail;
    Node* cur;
    if(n==0) return head=NULL;
    head->val=arr[0];
    head->next=NULL;
    tail=head;
    for(i=1;i<n;++i)
    {
        cur=(Node*)malloc(sizeof(Node));
        cur->val=arr[i];
        cur->next=NULL;
        tail->next=cur;
        tail=cur;
    }
    return head;
}

```

```
//程序终止后会自动销毁，这里额外显式销毁
```

```
void freeList(Node* head)
{
    while(head)
    {
        Node* tmp = head;
        head = head->next;
        free(tmp);
    }
}
```

```
int main() {
    int arr1[] = {1, 3, 5};
    int arr2[] = {2, 4, 6};
    Node* list1 = array_to_list(arr1, 3);
    printList(list1);
    Node* list2 = array_to_list(arr2, 3);
    printList(list2);
    Node* merged = mergeSortedLists(list1, list2);
```

```
    printList(merged);
    freeList(merged);
    return 0;
}
```

```
66    }
67
68 //程序终止后会销毁，这里额外显式销毁
69 void freeList(Node* head)
70 {
71     while(head)
72     {
73         Node* tmp = head;
74         head = head->next;
75         free(tmp);
76     }
77 }
78
```

问题 输出 调试控制台 终端 端口 2

+ ×  cppdbg: MergeSortedT1

```
*****
1 3 5
*****
2 4 6
*****
1 2 3 4 5 6
[1] + Done                                     "/usr/bin/gdb" --interpreter=mi --tty=
0<"/tmp/Microsoft-MIEngine-In-2os3n235.n5d" 1>"/tmp/Microsoft-MIEngine-
a.05k"
chi_shan@localhost:/mnt/d/FDU_1/CS10004_Programming$
```

模仿平衡树合并，递归地去做

2.

```
#include<stdio.h>
#include<stdlib.h>
#define max(a,b) ((a)>(b)?(a):(b))
#define min(a,b) ((a)<(b)?(a):(b))
struct Node
{
    int val;
    struct Node *nxt;
};
typedef struct Node Node;
Node *create_Node(int val)
{
```

```

Node *newNode=(Node *)malloc(sizeof(Node));
newNode->val=val;
newNode->nxt=NULL;
return newNode;
}

void append(Node *head,int val)
{
    Node *newNode=create_Node(val);
    Node *tail;
    tail=head;
    while(tail->nxt!=NULL)
    {
        tail=tail->nxt;
    }
    tail->nxt=newNode;
    newNode->nxt=NULL;
}

void delte_mn(Node *head)
{
    Node *tail;
    Node *prev;
    int mn;
    tail=head;
    mn=1000;
    while(tail->nxt!=NULL)
    {
        tail=tail->nxt;
        mn=min(mn,tail->val);
    }
    tail=head;
    prev=head;
    while(tail->nxt!=NULL)
    {
        tail=tail->nxt;
        if(tail->val==mn)
        {
            prev->nxt=tail->nxt;
            free(tail);
            return;
        }
        prev=tail;
    }
}

void delte_mx(Node *head)
{
    Node *tail;
    Node *prev;
    int mx;

```

```

tail=head;
mx=-1000;
while(tail->nxt!=NULL)
{
    tail=tail->nxt;
    mx=max(mx,tail->val);
}
tail=head;
prev=head;
while(tail->nxt!=NULL)
{
    tail=tail->nxt;
    if(tail->val==mx)
    {
        prev->nxt=tail->nxt;
        free(tail);
        return;
    }
    prev=tail;
}
}

void print_list(Node *head)
{
    Node *tail;
    tail=head;
    printf("#####\n");
    while(tail->nxt!=NULL)
    { tail=tail->nxt;
        printf("%d \t",tail->val);

    }

}

int main()
{
    Node Head;
    int n,int val;

```

```

Head.nxt=NULL;
Head.val=404;
//无意义的表头
scanf("%d",&n);
for(int i=0;i<n;++i)
{
    scanf("%d",&val);
    append(&Head,val);
}
delte_mn(&Head);
delte_mx(&Head);

```

```
    print_list(&Head);  
  
    return 0;  
}
```

The screenshot shows a C IDE interface with several tabs at the top: "MergeSortedT1.c", "DeleteMinMax.c" (selected), "divide&conquerT3.c", and "partition\_". The main area displays the "DeleteMinMax.c" code:

```
code_pack > DeleteMinMax.c > main()
83 void print_list(Node *head)
88     while(tail->nxt!=NULL)
92 }
93 }
94 }
95 int main()
96 {
97     Node Head;
98     int n;int val;
99
100    Head.nxt=NULL;
101    Head.val=404;
102    //无意义的表头
103    scanf("%d",&n);
104    for(int i=0;i<n;++i)
105    {
106        scanf("%d",&val);
107        append(&Head,val);
```

Below the code editor are tabs for "问题", "输出", "调试控制台", "终端", and "端口 2" (selected). The "输出" tab shows the following terminal output:

```
6
2
4
6 1 3 5
#####
2      4      3      5      [1] + Done          "/usr/bin/gdb" --
interpreter=mi --tty=${DbgTerm} 0</tmp/Microsoft-MIEngine-In-piegsaco.duf" 1>/tmp/Microsoft-MIEngine-Out-aui3e241.vno"
```

简单地分步开始写

3

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

#define max(a,b) ((a)>(b)?(a):(b))
#define min(a,b) ((a)<(b)?(a):(b))

struct Node
{
    int val;
    struct Node *pre;
    struct Node *nxt;
};

typedef struct Node Node;

void printList(Node* head,Node *tail
```

```

{
    Node* p = head;
    printf("*****\n");
    while(p!=tail)
    {
        printf("%d,", p->val);
        p = p->nxt;
    }

    printf("%d\n",tail->val);
}

void array_to_list(Node *head,Node *tail,int* arr, int n)
{
    Node *last,*cur;
    int i;

    head->nxt=tail;
    tail->pre=head;

    head->pre=NULL;
    tail->nxt=NULL;

    last=head;
    for(i=0;i<n;++i)
    {
        cur=(Node*)malloc(sizeof(Node));
        //
        cur->val=arr[i];
        cur->nxt=NULL;
        cur->pre=last;

        //更新
        last->nxt=cur;
        last=cur;
    }
    last->nxt = tail;
    tail->pre=last;
}

/*
有必要分治嘛
*/
Node * merge(Node *head1,Node *tail1,Node *head2,Node *tail2,int key)
{

    Node *cur1 = NULL; Node *cur2 =NULL;
    Node *cur3 = NULL; Node *cur4 =NULL;
    Node *p, *next;

    // 遍历第一段 [head1..tail1]
}

```

```

p = head1;
while (1) {
    next = p->nxt;
    p->nxt = p->pre = NULL;
    if (p->val < key) {
        if (!cur1) cur1 = cur2 = p;
        else { cur2->nxt = p; p->pre = cur2; cur2 = p; }
    } else {
        if (!cur3) cur3 = cur4 = p;
        else { cur4->nxt = p; p->pre = cur4; cur4 = p; }
    }
    if (p == tail1) break;
    p = next;
}

```

```

// 遍历第二段 [head2..tail2]
p = head2;
while (1) {
    next = p->nxt;
    p->nxt = p->pre = NULL;
    if (p->val < key) {
        if (!cur1) cur1 = cur2 = p;
        else { cur2->nxt = p; p->pre = cur2; cur2 = p; }
    } else {
        if (!cur3) cur3 = cur4 = p;
        else { cur4->nxt = p; p->pre = cur4; cur4 = p; }
    }
    if (p == tail2) break;
    p = next;
}

```

```

// 拼接
if (!cur1) {
    if (cur4) cur4->nxt = NULL;
    return cur3;
}
cur2->nxt = cur3;
if (cur3) cur3->pre = cur2;
if (cur4) cur4->nxt = NULL;
return cur1;

```

```

}

Node* solve(Node *head,Node *tail,int len,int key)
{
    Node *cur1 = NULL;
    Node *cur2 = NULL;

```

```

Node *cur3 = NULL;
Node *cur4 = NULL;
Node *p = head;
Node *next;

if (Len <= 0 ) return head;
if(head==NULL) return head;
while (1) {
    next = p->nxt;
    p->nxt = p->pre = NULL;
    if(p->val<key)
    {
        if(!cur1)cur1=cur2=p;
        else
        {
            cur2->nxt=p;
            p->pre=cur2;
            cur2=p;
        }
    }
    else
    {
        if (!cur3) cur3 = cur4 = p;
        else
        {
            cur4->nxt = p;
            p->pre =cur4;
            cur4 = p;
        }
    }
    if(p==tail)break;
    p=next;
}
if(!cur1)
{
    if (cur4)cur4->nxt=NULL;
    return cur3;
}
cur2->nxt=cur3;
if(cur3)cur3->pre=cur2;
if(cur4)cur4->nxt=NULL;
return cur1;
}

int main()
{
    int arr[10],n,key;
    Node Head,Tail;
    /*
     : 输入: 链表节点数n=6, 链表中的值1, 4, 3, 2, 5, 2, 分割值x=3; 输出: 1, 2, 2, 4, 3, 5。
    */
}

```

输入: 链表节点数n=5, 链表中的值2, 1, 2, 2, 1, 分割值x=2; 输出: 1, 1, 2, 2, 2。

```
/*
arr[0]=1,arr[1]=4,arr[2]=3,arr[3]=2,arr[4]=5,arr[5]=2;
n=6;
key=3;
array_to_list(&Head,&Tail,arr,n);
```

```
printList(Head.nxt,Tail.pre);
```

```
Node *new_head = solve(Head.nxt, Tail.pre, n, key);
Head.nxt = new_head;

Node *t = Head.nxt;
if(t)
{
    while (t->nxt)t=t->nxt;
    t->nxt=&Tail;
    Tail.pre = t;
}
else
{
    Head.nxt = &Tail;
    Tail.pre = &Head;
}
printList(Head.nxt,Tail.pre);
```

```
arr[0]=2,arr[1]=1,arr[2]=2,arr[3]=2,arr[4]=1;
```

```
n=5;
return 0;
}
```

```
11 //打印链表
12
13 void printList(Node* head,Node *tail)
14 {
15     Node* p = head;
16     printf("*****\n");
17     while(p!=tail)
18     {
19         printf("%d,", p->val);
20         p = p->nxt;
21     }
22
23     printf("%d\n",tail->val);
24 }
25 void array_to_list(Node *head,Node *tail,int* arr, int n)
26 {
*****  
1,4,3,2,5,2  
*****  
1,2,2,4,3,5  
[1] + Done          "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Micr  
.4qx" 1>"/tmp/Microsoft-MIEngine-Out-0xezezfsd3.yj3"  
chi_shan@localhost:/mnt/d/FDU_1/CS10004_Programming$
```

nf.c  
nf.exe  
al\_operations