

第 1 题. //P45 第 5 题，删除单链表中介于 min-max 之间的结点

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//单链表结构类型定义
typedef int datatype;
typedef struct node
{
    datatype data;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
void delete(linklist *, datatype, datatype);
int main()
{
    linklist*head;
    int min,max;
    head=create( );
    printf("原链表为: \n");
    print(head);
    puts ("*****请为 min 输入一个整数*****");
    scanf ("%d", &min);
    puts ("*****请为 max 输入一个整数*****");
    scanf ("%d", &max);
    delete (head, min, max);//调用单链表删除函数
    printf("*****删除介于 min 和 max 之间的结点后的链表*****\n");
    print(head);
    return 0;
}
```

测试用例 1:

输入: 2 6 9 10 15 23 38 45 50 56 67 89 100 136 138

min: 23

max:99

输出: 2 6 9 10 15 23 100 136 138

测试用例 2:

输入: 3 5 8 19 25 34 38 49 60 83 120 150 180

min:80

max:200

输出: 3 5 8 19 25 34 38 49 60

测试用例 3:

输入: 3 5 8 19 25 34 38 49 60 83 120 150 180

min:1

max:50

输出：60 83 120 150 180

测试用例 4:

输入：3 5 8 19 25 34 38 49 60 83 120 150 180

min:190

max:500

输出：3 5 8 19 25 34 38 49 60 83 120 150 180

第 2 题. // P45 第 7 题按照字符类型分解单链表

```
#include "stdafx.h"
#include<stdio.h>
#include<malloc.h>
typedef char datatype;
typedef struct node
{
    datatype data;
    struct node *next;
}linklist;

linklist* create( );
void resolve(linklist*,linklist*,linklist*,linklist*);
void print1(linklist*);
void print2(linklist*);
int main( )
{
    linklist *head,*letter,*digit,*other;
    head=create( );
    printf("*****原链表为*****\n");
    print1(head);
    letter=(linklist*)malloc(sizeof(linklist));//建立 3 个空循环链表
    letter->next=letter;
    digit=(linklist*)malloc(sizeof(linklist));
    digit->next=digit;
    other=(linklist*)malloc(sizeof(linklist));
    other->next=other;
    resolve(head, letter, digit, other);//调用分解单链表的函数
    printf("*****分解后的字母链表为*****\n");
    print2(letter);//输出循环链表
    printf("*****分解后的数字链表为*****\n");
    print2(digit);
    printf("*****分解后的其它字符链表为*****\n");
    print2(other);
    return 0;
}
```

测试用例 1:

输入:

dgjakdg*&?,8543246dghj

输出:

分解后的字母链表为: dgjakdgdghj

分解后的数字链表为: 8543246

分解后的其它字符链表为: *&?,

测试用例 2:

输入:

&%#dgj*#34akdg*&3246

输出:

分解后的字母链表为:dgjakdg

分解后的数字链表为: 343246

分解后的其它字符链表为: &%*##*&

测试用例 3:

输入:

7&%8dgj*kk#34ak@dg*&6

输出:

分解后的字母链表为: dgjkkakdg

分解后的数字链表为: 78346

分解后的其它字符链表为: &%*#@*&

第3题. //P74 第2题判断字符串是否是回文（中心对称）

```
#include "stdafx.h"
#include<stdio.h>
#include<malloc.h>
#include<string.h>
//定义字符串类型
#define maxsize 256
typedef struct
{
    char ch[maxsize];
    int len;
}seqstring;
seqstring * makestr();
void print(seqstring *);
int symmetry(seqstring *);//判字符串是否中心对称的函数声明

int main()
{
    seqstring *str;
    printf("请初始化字符串: ");
    str = makestr();
    if (symmetry(str)) printf("\n判定结果: 该字符串\"%s\"是回文\n\n", str->ch);
    else printf("\n判定结果: 该字符串\"%s\"不是回文\n\n", str->ch);
    return 0;
}
```

测试用例 1:

输入:

abdkgdkg

输出:

判定结果: 该字符串"abdkgdkg"不是回文

测试用例 2:

输入:

abdkgdk

输出:

判定结果: 该字符串"abdkgdk"不是回文

测试用例 3:

输入:

abdkkdba

输出:

判定结果：该字符串"abdkkdba"是回文

测试用例 4:

输入：

abdkukdba

输出：

判定结果：该字符串"abdkukdba"是回文

第 4 题.//朴素的模式匹配追踪算法（BF）测试

以类似下图的形式展示结果：

```
C:\WINDOWS\system32\cmd.exe
*****输入目标串：abcbabcbabckka
*****输入模式串：abcbabc
匹配成功！比较次数为：18
返回第一次匹配成功的位置（首字母所在下标）：5
请按任意键继续. . .
```

测试用例 1：

输入：

T:abcbabcbabc

P:abcbabc

输出：

匹配成功！比较次数为：18

返回第一次匹配成功的位置(首字母位序)： 6

测试用例 2：

输入：

T:abcbabcbabckka

P:abcbabc

输出：

匹配成功！比较次数为：18

返回第一次匹配成功的位置(首字母位序)： 6

测试用例 3：

输入：

T:abcbabcbabckka

P:abcabd

输出：

匹配失败！比较次数为：29

测试用例 4：

输入：

T:aaaaaaaaaaaaaaaaaakuu

P:aaaaak

输出：

匹配成功！比较次数为：90

返回第一次匹配成功的位置(首字母位序)： 15

测试用例 5：

输入:

T:aaaaaaaaaaaaaaaaaak

P:aaaaau

输出:

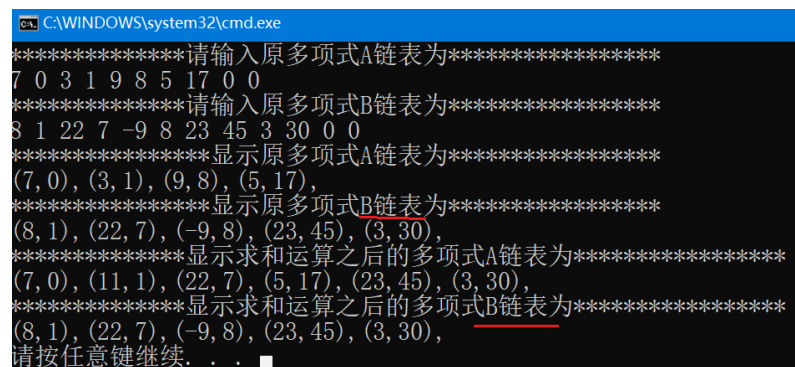
匹配失败! 比较次数为: 105

5. //多项式求和运算：设单链表 A 和 B 分别存储不同的多项式，要求完成多项式的求和运算，求和结果存放在 A 表中（备注：B 表不变，测试用例至少测 3 组不同情况）。

提示：

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//多项式单链表结构类型定义
typedef struct node
{
    int coef;
    int exp;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
void SumofPoly(linklist *, linklist *);
void main( )
{
    linklist*A, *B;
    printf("*****请输入原多项式A链表为*****\n");
    A = create( );
    printf("*****请输入原多项式B链表为*****\n");
    B = create( );
    printf("\n*****显示原多项式A链表为*****\n");
    print(A);
    printf("\n*****显示原多项式B链表为*****\n");
    print(B);
    SumofPoly(A, B); //调用多项式求和的函数
    printf("\n*****显示求和运算之后的多项式A链表为*****\n");
    print(A);
    printf("\n*****显示求和运算之后的多项式B链表为*****\n");
    print(B);
}
```

参考类似截屏。



```
C:\WINDOWS\system32\cmd.exe
*****请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
*****请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0
*****显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
*****显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
*****显示求和运算之后的多项式A链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
*****显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
请按任意键继续. . .
```

测试用例：

(1) $A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;

$$B(x)=8x+22x^7-9x^8-4x^{18}+30x^{25}+10x^{35}+19x^{55};$$

运行结果：

$$A(x)=7+11x+22x^7+5x^{17}-4x^{18}+2x^{20}+30x^{25}+10x^{35}+19x^{55};$$

$$B(x)=8x+22x^7-9x^8-4x^{18}+30x^{25}+10x^{35}+19x^{55};$$

测试用例：

$$(2) A(x)=19+3x+72x^7+6x^{17}+2x^{28}+10x^{35}+19x^{55};$$

$$B(x)=8x^4+22x^7-6x^{17}-2x^{28}$$

运行结果：

$$A(x)=19+3x+8x^4+94x^7+10x^{35}+19x^{55};$$

$$B(x)=8x^4+22x^7-6x^{17}-2x^{28}$$

测试用例：

$$(3) A(x)=23+3x+7x^6+16x^{18}+2x^{23}+10x^{32};$$

$$B(x)=-23-3x+17x^6-16x^{18}-2x^{23}-10x^{32};$$

运行结果：

$$A(x)=24x^6;$$

$$B(x)=-23-3x+17x^6-16x^{18}-2x^{23}-10x^{32};$$

测试用例：

$$(4) A(x)=23+3x+7x^6+16x^{18}+2x^{23}+10x^{32};$$

$$B(x)=6x^{12}+16x^{38}-2x^{42}-10x^{62};$$

运行结果：

$$A(x)=23+3x+7x^6+6x^{12}+16x^{18}+2x^{23}+10x^{32}+16x^{38}-2x^{42}-10x^{62};$$

$$B(x)=6x^{12}+16x^{38}-2x^{42}-10x^{62};$$

6. //多项式求和运算：设单链表 A 和 B 分别存储不同的多项式，要求完成多项式的求和运算，求和结果存放在 C 表中（备注：A 表、B 表均不变，生成新的多项式和链表 C；测试用例至少测 3 组不同情况）。

提示：

```
#include "stdafx.h"
#include<malloc.h>
#include<stdio.h>
//多项式单链表结构类型定义
typedef struct node
{
    int coef;
    int exp;
    struct node *next;
}linklist;
linklist* create( );
void print(linklist *);
linklist * SumofPoly(linklist *, linklist *);
void main( )
{
    linklist*A, *B,*C;
    printf("*****请输入原多项式A链表为*****\n");
    A = create( );
    printf("*****请输入原多项式B链表为*****\n");
    B = create( );
    printf("\n*****显示原多项式A链表为*****\n");
    print(A);
    printf("\n*****显示原多项式B链表为*****\n");
    print(B);
    C=SumofPoly(A, B); //调用多项式求和的函数
    printf("\n*****显示求和运算之后的多项式A链表为*****\n");
    print(A);
    printf("\n*****显示求和运算之后的多项式B链表为*****\n");
    print(B);
    printf("\n*****显示求和运算之后的多项式之和C链表为*****\n");
    print(C);
}
```

参考类似截屏：

```
C:\WINDOWS\system32\cmd.exe

*****请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
*****请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0
*****显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
*****显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
*****显示求和运算之后的多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),
*****显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),
*****显示求和运算之后的多项式C链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
请按任意键继续. . .
```

```
C:\WINDOWS\system32\cmd.exe

*****请输入原多项式A链表为*****
7 0 3 1 9 8 5 17 0 0
*****请输入原多项式B链表为*****
8 1 22 7 -9 8 23 45 3 30 0 0

*****显示原多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),

*****显示原多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),

*****显示求和运算之后的多项式A链表为*****
(7, 0), (3, 1), (9, 8), (5, 17),

*****显示求和运算之后的多项式B链表为*****
(8, 1), (22, 7), (-9, 8), (23, 45), (3, 30),

*****显示求和运算之后的多项式C链表为*****
(7, 0), (11, 1), (22, 7), (5, 17), (23, 45), (3, 30),
请按任意键继续. . .
```

```
C:\WINDOWS\system32\cmd.exe

*****请输入原多项式A链表为*****
3 56 23 60 34 70 -3 80 4 90 0 0
*****请输入原多项式B链表为*****
3 80 5 82 0 0

*****显示原多项式A链表为*****
(3, 56), (23, 60), (34, 70), (-3, 80), (4, 90),

*****显示原多项式B链表为*****
(3, 80), (5, 82),

*****显示求和运算之后的多项式A链表为*****
(3, 56), (23, 60), (34, 70), (-3, 80), (4, 90),

*****显示求和运算之后的多项式B链表为*****
(3, 80), (5, 82),

*****显示求和运算之后的多项式C链表为*****
(3, 56), (23, 60), (34, 70), (5, 82), (4, 90),
请按任意键继续. . .
```

测试用例：

(1) $A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;
 $B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55}$;

运行结果：

$A(x) = 7 + 3x + 9x^8 + 5x^{17} + 2x^{20}$;
 $B(x) = 8x + 22x^7 - 9x^8 - 4x^{18} + 30x^{25} + 10x^{35} + 19x^{55}$;
 $C(x) = 7 + 11x + 22x^7 + 5x^{17} - 4x^{18} + 2x^{20} + 30x^{25} + 10x^{35} + 19x^{55}$;

测试用例：

$$(2) \begin{aligned} A(x) &= 19 + 3x + 72x^7 + 6x^{17} + 2x^{28} + 10x^{35} + 19x^{55}; \\ B(x) &= 8x^4 + 22x^7 - 6x^{17} - 2x^{28}; \end{aligned}$$

运行结果：

$$\begin{aligned} A(x) &= 19 + 3x + 72x^7 + 6x^{17} + 2x^{28} + 10x^{35} + 19x^{55}; \\ B(x) &= 8x^4 + 22x^7 - 6x^{17} - 2x^{28}; \\ C(x) &= 19 + 3x + 8x^4 + 94x^7 + 10x^{35} + 19x^{55}; \end{aligned}$$

测试用例：

$$(3) \begin{aligned} A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\ B(x) &= -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32}; \end{aligned}$$

运行结果：

$$\begin{aligned} A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\ B(x) &= -23 - 3x + 17x^6 - 16x^{18} - 2x^{23} - 10x^{32}; \\ C(x) &= 24x^6; \end{aligned}$$

测试用例：

$$(4) \begin{aligned} A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\ B(x) &= 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62}; \end{aligned}$$

运行结果：

$$\begin{aligned} A(x) &= 23 + 3x + 7x^6 + 16x^{18} + 2x^{23} + 10x^{32}; \\ B(x) &= 6x^{12} + 16x^{38} - 2x^{42} - 10x^{62}; \\ C(x) &= 23 + 3x + 7x^6 + 6x^{12} + 16x^{18} + 2x^{23} + 10x^{32} + 16x^{38} - 2x^{42} - 10x^{62}; \end{aligned}$$