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"abdkgdkg"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

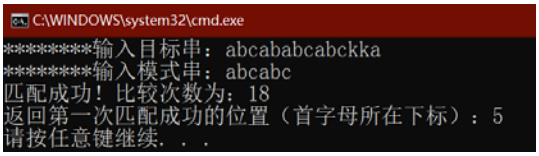
测试用例 4:

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：



测试用例 1：

输入:

T:abcababcabc

P:abcabc

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:abcababcabckka

P:abcabc

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:abcababcabckka

P:abcabd

输出:

匹配失败！比较次数为：29

测试用例 4：

输入:

T:aaaaaaaaaaaaaaaaaaakuu

P: aaaaak

输出:

匹配成功！比较次数为：90

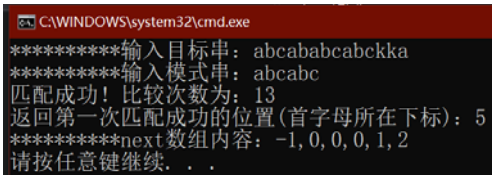
返回第一次匹配成功的位置(首字母位序)：15

测试用例 5：

输入:
T:aaaaaaaaaaaaaaaaaak
P:aaaaau
输出:
匹配失败！ 比较次数为：105

第 5 题. //改进的模式匹配追踪算法（KMP）测试

以类似下图的形式展示结果：



测试用例 1：

输入:
目标串: abcababcabc
模式串: abcabc

输出:
匹配成功！比较次数为：13
返回第一次匹配成功的位置(首字母位序)：5
**********next 数组内容：-1,0,0,0,1,2

测试用例 2：

输入:
目标串: abcababcabckka
模式串: abcabc

输出:
匹配成功！比较次数为：13
返回第一次匹配成功的位置(首字母位序)：5
**********next 数组内容：-1,0,0,0,1,2

测试用例 3：

输入:
目标串: abcababcabckka
模式串: abcabd

输出:
匹配失败！ 比较次数为：20
**********next 数组内容：-1,0,0,0,1,2

测试用例 4：

输入:
目标串: aaaaaaaaaaaaaaaaaakuu

模式串: aaaaak

输出:
匹配成功！比较次数为：34
返回第一次匹配成功的位置(首字母位序)：14
*********next 数组内容：-1,0,1,2,3,4

测试用例 5：

输入:
目标串: aaaaaaaaaaaaaaaaaak
模式串: aaaaau

输出:
匹配失败！ 比较次数为：40
*********next 数组内容：-1,0,1,2,3,4

第 6 题：

功能要求：单链表逆置（要求和第 1 次上机采用不同的算法实现，时间复杂度 O(n)）