实验 5-1：

从键盘输入一个长度为 N（比如 10）的整型数组，而后将数组中小于零的元素移动到数组的前端，大于零的元素移到数组的后端，等于零的元素留在数组中间。比如原来数组为：2 -5 -89 75 0 -89 0 93 48 0，经过处理后的数组为：-5 -89 -89 0 0 0 75 93 48 2。由于不要求数组有序，所以不允许用排序方法

源代码：

```
#include<stdio.h>
#include<stdlib.h>
#define  N 10
void  main()
{
     int  a[N]={2,-5,-89,75,0,-89,0,93,48,0};
     int  i,t,j;
     for(i=0;i<N;i++)
     {
          printf("%4d",a[i]);
     }
     printf("\n");
     for(i=0;i<N;i++)
     {
          for(j=i+1;j<N;j++)
          {
               if(a[i]>0&&a[j]<0)
               {
                    t=a[i];
                    a[i]=a[j];
                    a[j]=t;
               }
               else  if(a[i]>0&&a[j]==0)
               {
                    t=a[i];
                    a[i]=a[j];
                    a[j]=t;
               }
               else  if(a[i]==0&&a[j]<0)
               {
                    t=a[i];
                    a[i]=a[j];
                    a[j]=t;
               }
          }
     }
     for(i=0;i<N;i++)
          printf("%4d",a[i]);
```

```
    printf("\n");
    system("pause");
}
```

实验 5-2：

设数组 a 的定义如下：

int a[20] = {2,4,6,8,10,12,14,16}；已存入数组中的数据值已经按由小到大的顺序存放，现从键盘输入一个数据，把它插入到数组中，要求插入新数据以后，数组数据仍然保持有序。请编写一个程序实现上述功能

源代码：

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int a[9]={2,4,6,8,10,12,14,16};
    int i,j,number;
    printf("原数组\n");
    for(i=0;i<=8;i++)//输出数组原有的元素
    {
        printf("%d ",a[i]);
    }
    printf("\n");
    printf("请输入一个数据\n");
    scanf("%d",&number);

    i=0;//以下三行找到插入位置
    while(a[i]<number&&i<8)
    {
      i++;
     }
    for(j=7;j>=i;j--)//将插入点以后的元素顺序往后移一位
    {
      a[j+1]=a[j];
     }
    a[i]=number;//插入数据
    printf("输出改变后的数组\n");
    for(i=0;i<=8;i++)//输出插入后仍有序的数组
    {
        printf("%d ",a[i]);
    }
    system("pause");
}
```

实验 5-3：

写一个 3 x 5 矩阵的转置程序，输出其原矩阵的值和转置以后的结果

源代码：

```
#include<stdio.h>
#include<stdlib.h>
main()
{
    int i,j;
    int a[3][5]={1,2,3,4,5,6,7,8,9,4,5,6,7,8,9};
    for(i=0;i<3;i++)
    {
        for(j=0;j<5;j++)
        {
            printf("%d  ",a[i][j]);
        }
        printf("\n");
    }
     printf("转置后的矩阵为\n");
    for(i=0;i<5;i++)
    {
       for(j=0;j<3;j++)
       {
           printf("%d  ",a[j][i]);
       }
         printf("\n");
    }
    system("pause");
}
```

实验 6-1：

写一个函数 int digit(int n , int k)，它返回数 n 的从右向左的第 k 个十进数字值。例如，函数调用 digit(1234,2)将返回值 3

源代码：

```
#include<stdio.h>
#include<stdlib.h>
int digit(int n,int k)
{
    int i,j,a,b;
    b=n%10;
    a=(n/10)%10;
    j=(n/100)%10;
    i=(n/1000)%10;
    n=1000*i+100*j+10*a+b;
```

```
    if(k==1)
       printf("%d",b);
    else if(k==2)
       printf("%d",a);
    else if(k==3)
       printf("%d",j);
    else if(k==4)
       printf("%d",i);
    }
main()
{
       int x,y;
       printf("请输入一个数据\n");
       scanf("%d%d",&x,&y);
       digit(x,y);
       system("pause");
}
```

实验 6-2：

写一个函数 reverse(char s[])，将字符串 s[]中的字符串倒序输出。试分别用递归和非递归两种形式编写

源代码：

递归：

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void res(char s[],int len)
{
    if(len>0)
    {
        printf("%c",s[len-1]);
        len--;
        res(s,len);
    }
}
void reverse(char b[])
{
    res(b,strlen(b));
}
void main()
{
    char a[100];
```

```
    gets(a);
    reverse(a);//或者 res(a,strlen(a));
    system("pause");
}
```

源代码：

非递归：

```
#include<stdio.h>
#include<stdlib.h>
void reverse(char  s[]);
void main()
{
    char s[100];
    gets(s);
    reverse(s);
}
void reverse(char  s[])     //非递归调用的 reverse 函数
{
    int j,i=0;
    do
    {
        i++;
    }while(s[i]!='\0');
    for(j=i-1;j>=0;j--)
    {
        printf("%c",s[j]);
    }
    printf("\n");
    system("pause");
}
```

实验 6-3：

一个数如果从左到右和从右到左读，数字是相同的，则称这个数字为回文数，比如 898、1221、15651 都是回文数。求：既是回文数又是质数的 5 位十进制数有多少个？要求：回文判断和质数判断都需要通过子函数实现，输出的时候要求 5 个数字一行

源代码 1：

```
#include"stdio.h"
#include  "stdafx.h"
int hw(int  n)
{
    int  a[5],i,k=n,j;
    for(i=2;i<n/2;i++)
```

```
    {
        if(n%i==0)
            return 0;
    }
    for(i=0;i<5;i++)
    {
        a[i]=k%10;
        k=k/10;
    }
    if(n==a[0]*10000+a[1]*1000+a[2]*100+a[3]*10+a[4])
        return 1;
    else
        return 0;
}
int main()
{
    int i,s,j=0;
    for(i=10000;i<=99999;i++)
    {
        s=hw(i);
        if(s)
        {
            printf("%10d",i);
            j++;
            if(j%5==0)
                printf("\n");
        }
    }
    return 0;
}
```

源代码 2：

```
#include"stdio.h"
int hw(int n)
{
    int a[5],i,k=n,j;
    for(i=0;i<5;i++)
    {
        a[i]=k%10;
        k=k/10;
    }
    if(n==a[0]*10000+a[1]*1000+a[2]*100+a[3]*10+a[4])
    {
```

```
        for(i=2;i<n/2;i++)
    {
        if(n%i==0)
            return 0;
    }

     return 1;
    }

    return 0;
}
int main()
{
    int i,s,j=0;
    for(i=10000;i<=99999;i++)
    {
        s=hw(i);
        if(s)
        {
            printf("%10d",i);
            j++;
            if(j%5==0)
                printf("\n");
        }
    }
    return 0;
}
```

比较上面两个代码的区别。

实验 7-1：

编写一个函数 char *delk(char *sp)，把 sp 所指向的字符串中所有的“$”字符删除，并把处理后的字符串指针返回

源代码：

```
#include <stdio.h>
char *delk(char *sp)
{

    char *p=sp,*q=sp;
    while(*p)
    {
        if(*p++!='$')
        {
            q++;
        }
```

```
            *q=*p;
        }
        return sp;
}
int main()
{
        char a[100];
        gets(a);
        printf("%s\n",delk(a));
        return 0;
}
```

实验 7-2：

写一个函数 int find(char *s1, char *s2)，函数 find 的功能是查找串 s1 中是否包含指定的词（s2 指向），如果存在则返回第 1 次出现的位置，否则返回-1.约定串中的词由 1 个或 1 个以上的空格符分隔。

源代码：

```
#include <stdio.h>

int find(char *s1, char *s2)
{
        int count = 0;
        int i = 0;
        while( *(s1+count)!='\0' )
        {
            if  ( (*(s1+count)==*s2) && ((*(s1+count-1)==' ')||(count==0) ))
            {
                while( (*(s2+i)!='\0')&&(*(s1+count+i)!='\0') )
                {
                    if( *(s2+i)==*(s1+count+i) )
                    {
                        i++;
                    }
                    else
                    {
                        break;
                    }
                }

                if( (*(s2+i)=='\0')&& ( (*(s1+count+i)==' ')|| (*(s1+count+i)=='\0')))
                {
                    return count+1;
                }
            }
```

```
            count++;
        }

        return -1;
}
int main()
{
        char *s1="I am a girl",*s2="girl";
        printf("%d",find(s1,s2));
        printf("\n");
        return 0;
}
```

实验 7-3：

定义函数 void Merge(int a[], int n, int b[], int m)，参数 a、b 为一维数组，数组中的数据为升序排列，n 和 m 分别为它们的元素个数。函数的功能为：将数组 a 和 b 合并为一个数组，合并后的结果存放于数组 a 中，要求合并后的数组 a 仍旧为升序排列。请编程实现，并编写 main 函数对其测试。

源代码：

```
#include<stdio.h>
void Merge(int a[],int n,int b[],int m)
{
    int *p,*q,i,j,k;
    p=a;
    q=b;


    for(i=0;i<10;i++)
    {
        if(i>0&&*(p+i)<*(p+i-1))
        {
            *(p+i)=*q;
             q++;
        }
    }
    for(i=0;i<10;i++)
    {
       for(j=i+1;j<10;j++)
      {
            if(*(p+i)>*(p+j))
            {
                k=*(p+i);
                *(p+i)=*(p+j);
```

```
                *(p+j)=k;
            }
        }
    }
}
void main()
{
   int a[10]={2,3,5,7,9},b[5]={1,4,6,8,10};
   int i;
   Merge(a,10,b,5);
   for(i=0;i<10;i++)
      printf("%3d",a[i]);
 }
```