作业五

1. 计算下列串的 next 数组：

（1）"ABCDEFG"

-1 0 0 0 0 0 0

（2）"AAAAAAAA"

-1 0 1 2 3 4 5 6

（3）"BABBABAB"

-1 0 0 1 1 2 3 2

（4）"AAAAAAB"

-1 0 1 2 3 4 5

（5）"ABCABDAAABC"

-1 0 0 0 1 2 0 1 1 1 2

（6）"ABCABDABEABCABDABF"

-1 0 0 0 1 2 0 1 2 0 1 2 3 4 5 6 7 8

（7）"ABBACXY"

-1 0 0 0 1 0 0

2. 要求输入两个字符串 s 和 t，统计 s 包含串 t 的个数。

```
void GetNext(char* p, int* nextarr)
{
    nextarr[0] = -1;
    int j = 0;
    int k = -1;
    int len = strlen(p);
    while (j < len - 1)
    {
        if (k == -1 || p[j] == p[k])
            nextarr[++j] = ++k;
        else
            k = nextarr[k];
    }
}
int KMP(char* s, char* p)
{
    int i = 0;
    int j = 0;
    int slen = strlen(s);
    int plen = strlen(p);
    int* nextarr = new int[plen];
    GetNext(p, nextarr);
    while (i < slen && j < plen)
    {
```

```
        if (j == -1 || s[i] == p[j])
        {
            i++;
            j++;
        }
        else
        {
            j = nextarr[j];
        }
    }
    if (j == plen)
        return i - j;
    else
        return -1;
}
int NumOfSubstr(char* ss, char* p)
{
    int sum = 0;
    int pos = 0;
    int plen = strlen(p);
    char* s = new char[strlen(ss) + 1];
    strcpy(s, ss);//避免损失原串
    while (1)
    {
        pos = KMP(s, p);
        if (pos != -1)
        {
            sum++;
            strcpy(s, s + pos + plen);
        }
        else
            break;
    }
    return sum;
}
```

3. 编写从串 s 中删除所有与串 t 相同的子串的算法

```
void DeleteAllSubstr(char* s, char* p)
{
    int pos = 0;
    int plen = strlen(p);
    while (1)
    {
        pos = KMP(s, p);//KMP重复部分不再粘贴
        if (pos != -1)
```

```
            strcpy(s + pos, s + pos + plen);
        else
            break;
    }
}
```

4. 试给出求串 s 和串 p 的最大公共子串的算法

```
string longestCommonSubstring(const string& s1, const string& s2)
{
    int len1 = s1.length();
    int len2 = s2.length();
    int start1 = -1;
    int start2 = -1;
    int longest = 0;
    for (int i = 0; i < len1; ++i)
    {
        for (int j = 0; j < len2; ++j)
        {
            int len = 0;
            for (int m = i, n = j; m < len1 && n < len2;)
            {
                if (s1[m] == s2[n])
                    m++, n++, len++;
                else
                    break;
            }
            if (len > longest)
            {
                longest = len;
                start1 = i;
                start2 = j;
            }
        }
    }
    if (longest == 0)
        return "";
    return s1.substr(start1, longest);
}
```

5. 编写一个函数来颠倒单词在字符串里的出现顺序。例如，把字符串"Do or do not, there is no try. "转换为"try. no is there not do, or Do"。假设所有单词都以空格为分隔符，标点符号也当做字母来对待。请对你的设计思路做出解释，并对你的解决方案的执行效率进行评估。

效率 O(n^2)，解释如注释

```
void ReverseWords(char* str)
{
    int i, j, k = 0;
    int len = strlen(str);
    char* tmp = new char[len + 1];
    int index = len - 1;//index记录复制的上界
    for (i = len - 1; i >= 0; i--)
    {
        if (str[i] == ' ')//遇到空格开始复制
        {
            for (j = i + 1; j <= index; j++)//从空格后一个开始复制到index
            {
                tmp[k++] = str[j];
            }
            tmp[k++] = ' ';
            index = i - 1;//改变index，使其到达下一个字母处
            i = index - 1;//i置于index前
        }
        if (i == 0)//遇到首位开始收尾
        {
            for (j = i; j <= index; j++)
                tmp[k++] = str[j];
            tmp[k] = 0;//末尾添加\0
        }
    }
    strcpy(str, tmp);//tmp内容移到原字符串
    delete []tmp;//释放tmp
}
```

6. 设有三对角矩阵 $A_{n\times n}$，将其按行优先顺序压缩存储于一维数组 $b[3*n-2]$中，使得 $a_{ij}=b[k]$，请用 k 表示 i，j 的下标变换公式。

   i=(k+1)/3;

   j=(k+1)%3+(k+1)/3-1;

7. 若在矩阵 A_{mxn} 中存在一个元素 a_{ij} $(0\leqslant i\leqslant m-1,\ 0\leqslant j\leqslant n-1)$满足：$a_{ij}$ 是第 i 行元素中最小值，且又是第 j 列元素中最大值，则称此元素值为该矩阵的一个马鞍点。假设以二维数组存储矩阵 A_{mxn}，试编写求出矩阵中所有马鞍点的算法。

```
void SaddlePoint(int** A,int M,int N)
{
    int *min=new int[M], *max=new int[N];
    int i, j;
    for (i = 0; i < M; i++) //求出每行最小数
    {
        min[i] = A[i][0];
        for (j = 1; j < N; j++)
            if (min[i] > A[i][j])
```

```
                min[i] = A[i][j];
    }
    for (j = 0; j < N; j++) //求出每列最大数
    {
        max[j] = A[0][j];
        for (i = 1; i < M; i++)
            if (max[j] < A[i][j])
                max[j] = A[i][j];
    }
    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            if (min[i] == max[j])//比较可求出
                cout << "[" << i << "]" << "[" << j << "] = " << A[i][j] << endl;
    }
```

8. 编写算法计算一个稀疏矩阵的对角线元素之和，要求稀疏矩阵用三元组顺序表表示。

```
template <class T>
struct Tri
{
    int row, col;
    T ele;
};
struct TSM
{
    struct Tri* data;
    int mu, nu, tu;//行，列，元素数
};
template <class T>
T DiagonalSum(struct TSM& M)
{
    T sum = 0;
    for (int i = 0; i < M.tu; i++)
        if (M.data[i].row == M.data[i].col)
            sum += M.data[i].ele;
    return sum;
}
```