数据结构 A
第二章参考答案

题 目 范 围 ：串
授 课 老 师 ：何璐璐
助 教 ：王思怡（撰写人），赵守玺
联 系 邮 箱 ：2021302111095@whu. edu. cn

目录

1. 单选题

题目1

以下关于串的叙述中正确的是 _____

A.串是一种特殊的线性表

B.串中元素只能是字母

C.空串就是空白串

D.串的长度必须大于零

答案：A

解析：串是一种特殊的线性表，其元素为单个字符，长度可以为0。

题目2

两个字符串相等的条件是 _____

A.串的长度相等

B.含有相同的字符集

C.串的长度相等并且对应的字符相同

D.串的长度和对应字符都一致并且存储结构相同

答案：C

解析：串相等比较的是数据内容而不是存储结构

题目3

若串S=“software”，其不同子串的个数 _____

A.37

B.36

C.8

D.9

答案：A

解析：

字串： n（n+1）/2 + 1

非空子串：n（n+1）/2

非空真子串：n（n+1）/2 - 1

题目4

在KMP模式匹配中用next数组存放模式串的部分匹配信息，当模式串位j与目标串i比较时两字符不相等，则i的位移方式是 _____

A.i=next[j]

B.i不变

C.i=0

D.i=i-j+1

答案：B

2. 编程题

2.1. 两串的最长相同前后缀

【问题描述】

给定两个字符串s1和s2，求最长的s1前缀ss使得ss为s2的最长后缀，输出该字符串和其长度。

【输入形式】

输入的每个测试用例由两行构成，第一行为s1，第二行为s2。假设所有输入的数据均为小写字母。

【输出形式】

每个测试用例的输出由单行组成，其中包含最长的字符串，该字符串是s1的前缀和s2的后缀，后面跟着该前缀的长度。如果最长的此类字符串是空字符串，则输出应为0。s1和s2的长度最多为50000。

【样例输入】

aaariemann

marjorieaaa

【样例输出】

aaa 3

【样例说明】

输入的第一行字符串s1=‘aaariemann’，第二行字符串s2=‘marjorieaaa’。s1的前缀和s2的后缀最长相等字符串为aaa，因此输出aaa 3，而不是a 1。测试数据存放在in. txt文件中。

【评分标准】

共10个测试用例，每通过一个测试得10分。

参考答案：

```cpp
#include<iostream>
#include <fstream>
#include<cstring>
using namespace std;
#define MAXN 100005
int nex[MAXN];
void GetNext(char *s)     //求 s 的 next 数组
{ int n=strlen(s);
    int j=0,k=-1;
    nex[0]=-1;
    while (j<n)        //含求 next[n]
    { if (k==-1 || s[k]==s[j])
        { k++; j++;
            nex[j]=k;
        }
        else k=nex[k];
    }
}
int main()
{
    ifstream inFile;
    inFile.open("in.txt", ios::in);
    if(!inFile){
        cout << "error open in.txt!" << endl;
    }

    char s1[MAXN],s2[MAXN];
    while (inFile >> s1 >> s2){
        inFile >> s1 >> s2;
        int n=strlen(s1);
        int m=strlen(s2);
        int minn=(n>m?m:n);
        strcat(s1,s2);      //s1 和 s2 连接起来存放在 s1 中
        GetNext(s1);
        int ans=nex[n+m];
        if(ans>minn) ans=minn;
        if (ans == 0 || ans == -1)
            printf("0\n");
        else
        { s1[ans] = 0;
            printf("%s %d\n", s1, ans);
        }
    }
    inFile.close();
    return 0;
}
```

解析：

本题目要求寻找两个给定字符串s1和s2中，s1的最长前缀，该前缀同时也是s2的最长后缀。解题的核心是应用字符串匹配算法中的Next数组。

为了找到s1的前缀和s2的后缀的最长匹配，可以将s1和s2连接成一个新的字符串，构建出一个可以同时考虑s2后缀和s1前缀的字符串结构。Next数组是KMP算法的核心部分，它记录了字符串中每一位置前的子串中，有多长的相同前缀和后缀。通过这个数组，我们可以快速找出两个字符串最长的公共前缀和后缀。

需要注意，我们需要保证这个长度不超过s1和s2中较短的那一个字符串的长度。同时，在使用Next数组时，需要注意数组索引的处理，避免越界。

2.2. 压缩字符串

【问题描述】

给定一组字符，使用原地算法将其压缩。压缩后的长度必须始终小于或等于原数组长度。数组的每个元素应该是长度为1的字符（不是int整数类型）。在完成原地修改输入数组后，返回数组的新长度。

【输入形式】

输入一组字符，字符间以空格隔开。

【输出形式】

输出一组字符，重复的字符用重复次数替代。每个数字在数组中都有它的位置。输出字符间以空格隔开。

【样例输入】

a a b b c c c

【样例输出】

a 2 b 2 c 3

【样例说明】

"aa"被"a2"替代。"bb"被"b2"替代。"ccc"被"c3"替代。测试数据存放在in. txt文件中。

【评分标准】

共10个测试用例，每通过一个测试得10分。

参考答案：

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>

using namespace std;
class Solution {
public:
    int compress(vector<char>& chars){
        string tmp;
        int i=0,k=0;        //i 用于遍历，k 表示压缩串的长度
        int cnt;
        while (i<chars.size()){
            chars[k]=chars[i];      //添加chars[i]字符
            k++;
            cnt=1;          //累计chars[i]相邻重复个数（平台长度）
            i++;
            while (i<chars.size() && chars[i]==chars[k-1]){
                i++;
                cnt++;
            }
            if (cnt>1){         //仅cnt>1 时进行压缩
                tmp=to_string(cnt);     //将cnt 整数转换为数字串
                for (int j=0;j<tmp.size();j++) { //将每个数字字符插入到chars
                    chars[k]=tmp[j];
                    k++;
                }
            }
        }
        return k;
    }
};
int main() {
    ifstream inFile;
    inFile.open("in.txt", ios::in);
    if(!inFile){
        cout << "error open in.txt!" << endl;
```

```
    }
    string s;
    getline(inFile, s);    // read a line
    char c;
    vector<char> str;
    istringstream strin(s);
    while (strin >> c){
        str.push_back(c);
    }

    Solution obj;
    int n;
    n = obj.compress(str);

    for(int i = 0; i < n; i++)
        cout << str[i] << ' ' ;
    cout << endl;
    inFile.close();
    return 0;
}
```

解析：

本题目的是实现一个字符串压缩算法，将连续出现的字符简化为字符后跟其重复次数。题目要求使用原地算法，即直接在输入数组上进行修改，并最终返回修改后的数组长度。

解题思路：使用两个指针 i 和 k，其中 i 用于遍历整个字符数组，k 用于在同一个数组中记录压缩结果的位置。另外一个变量 cnt 用来计算每个字符连续出现的次数。首先，将当前字符直接赋值到 chars[k]，然后将 k 加一以便记录下一个字符或计数。对于每个字符，通过内部循环计算连续出现的次数（cnt）。若当前字符与下一个字符相同，则 cnt 加一并更新 i 的值。当字符的连续出现次数大于1时，将这个次数转换成字符串，并将其每一个字符依次放入 chars[k]，同时更新 k 的值。遍历结束后，k 的值就是压缩后的数组长度。返回这个长度即可。

此题需要注意压缩必须在原数组上进行，意味着我们不能使用额外的数组来存储结果。同时，解题时需要特别注意不要越界写数据。