字符串处理

一、乙类题目：

1、写出这个数

读入一个正整数 *n*，计算其各位数字之和，用汉语拼音写出和的每一位数字。

输入格式：

每个测试输入包含 1 个测试用例，即给出自然数 *n* 的值。这里保证 *n* 小于 10100。

输出格式：

在一行内输出 *n* 的各位数字之和的每一位，拼音数字间有 1 空格，但一行中最后一个拼音数字后没有空格。

输入样例：

```
1234567890987654321123456789
```

输出样例：

```
yi san wu
```

核心代码：

```
int main() {
    int num[100] = {0}, sum = 0, n = 0;
    string str;
    getline(cin, str);
    int len = str.size();
    for(int i = 0; i < len; i++) {
        num[i] = (int)str[i] - '0';
        sum += num[i];
    }
    vector<int> c;
    do {
        c.push_back(sum % 10);
        sum = sum / 10;
    } while(sum != 0);
    for(int i = c.size()-1; i>=0 ; i--) {
        switch(c[i]){
            case 1: printf("yi"); break;
            case 2: printf("er"); break;
            case 3: printf("san"); break;
```

```
            case 4: printf("si"); break;
            case 5: printf("wu"); break;
            case 6: printf("liu"); break;
            case 7: printf("qi"); break;
            case 8: printf("ba"); break;
            case 9: printf("jiu"); break;
            case 0: printf("ling"); break;
        }
        if(i != 0) printf(" ");
    }
    return 0;
}
```

2、说反话

给定一句英语，要求你编写程序，将句中所有单词的顺序颠倒输出。

输入格式：

测试输入包含一个测试用例，在一行内给出总长度不超过 80 的字符串。字符串由若干单词和若干空格组成，其中单词是由英文字母（大小写有区分）组成的字符串，单词之间用 1 个空格分开，输入保证句子末尾没有多余的空格。

输出格式：

每个测试用例的输出占一行，输出倒序后的句子。

输入样例：

```
Hello World Here I Come
```

输出样例：

```
Come I Here World Hello
```

核心代码：

```
#include <iostream>
#include <vector>
#include <sstream>
using namespace std;

int main() {
    string str;
    getline(cin, str);

    stringstream ss(str);
    vector<string> words;               // 定义words向量，用于存储单词
    string word;
    while(ss >> word){                  // 使用 >> 操作符读取单词（以空格为分
隔符）
```

```
        words.push_back(word);
    }
    int len = words.size();
    for(int i=len-1; i>=0; i--) {
        cout << words[i];
        if(i != 0) cout << ' ';
    }
    return 0;
}
// 使用反向迭代器遍历向量：rebegin()、rend()
//for(auto it = words.rbegin(); it != words.rend(); ++it) {
//    cout << *it;
//    if(next(it) != words.rend()) // 判断是否为最后一个单词
//       cout << ' ';
//}
```

```
    cout<<a<<endl;
    cout<<b<<endl;
    cout<<c<<endl;
    cout<<d<<endl;
```

3、福尔摩斯的约会

大侦探福尔摩斯接到一张奇怪的字条：

```
我们约会吧！
3485djDkxh4hhGE
2984akDfkkkkggEdsb
s&hgsfdk
d&Hyscvnm
```

大侦探很快就明白了，字条上奇怪的乱码实际上就是约会的时间`星期四 14:04`：

- **前面两字符串**中：
  - 第 1 对相同的**大写英文字母**（大小写有区分）是第 4 个字母 `D`，代表星期四；
  - 第 2 对相同的**字符**是 `E` ，那是第 5 个英文字母，代表一天里的第 14 个钟头（于是一天的 0 点到 23 点由数字 0 到 9、以及大写字母 `A` 到 `N` 表示）；
- **后面两字符串**中：
  - 第 1 对相同的**英文字母** `s` 出现在第 4 个位置（从 0 开始计数）上，代表第 4 分钟。

输入格式：

现给定两对字符串，请帮助福尔摩斯解码得到约会的时间。输入在 4 行中分别给出 4 个非空、不包含空格、且长度不超过 60 的字符串。

输出格式：

在一行中输出约会的时间，格式为 `DAY HH:MM`，其中 `DAY` 是某星期的 3 字符缩写，即 `MON` 表示星期一，`TUE` 表示星期二，`WED` 表示星期三，`THU` 表示星期四，`FRI` 表示星期五，`SAT` 表示星期六，`SUN` 表示星期日。题目输入保证每个测试存在唯一解。

输入样例：

```
3485djDkxh4hhGE
2984akDfkkkkggEdsb
s&hgsfdk
d&Hyscvnm
```

输出样例：

```
THU 14:04
```

核心代码：

- **ASCII码表**：数字、小写字母、大写字母；
- 当字符数组的元素都是字符串时，如何定义；

```
#include <iostream>
using namespace std;

char Weekday[7][4] = {"MON", "TUE", "WED", "THU", "FRI", "SAT", "SUN"};
char Hour[24][3] = {"00","01","02","03","04","05","06","07","08","09","10",
                    "11","12","13","14","15","16","17","18","19","20","21",
                    "22","23"};

int main(){
    string a, b, c, d;
    char A, B;
    int num = 0, C = -1;
    getline(cin, a);
    getline(cin, b);
    getline(cin, c);
    getline(cin, d);

    for(int i=0;i<a.size() && i<b.size();i++){      //保证两个字符串均不越界
        if(a[i] == b[i]){
            //星期几，第一对相同的大写英文字母，注意：必须是：'A'~'G'之间的大写
字母!!!
            if((a[i]>=65 && a[i]<=71) && num ==0){
                A = a[i];num++;continue;
            }
            //几点钟，第二对相同字符：0~9、A~N
            if((a[i]>=48 && a[i]<=59) || (a[i]>=65 && a[i]<=78)){
                if(num == 1){
                    B = a[i];num++;break;
                }
```

```
            }
        }
    }
    for(int i=0;i<c.size()  && i<d.size();i++){      //保证两个字符串均不越界
        if(c[i] == d[i]){
            //几分钟，后两个字符串的第一对相同的英文字母出现的下标
            if((c[i]>=65 && c[i]<=90) || (c[i]>=97 && c[i]<=122)){
                C = i;break;
            }
        }
    }

    printf("%s ", Weekday[A-'A']);                          //星期几
    if(B>=48 && B<=59) printf("%s:", Hour[B-48]);           //0~9 小时
    else if(B>=65 && B<=78) printf("%s:", Hour[B-55]);      //10~23 小时
    printf("%02d", C);                                      //分钟
    return 0;
}
```

4、个位数统计：

1021 个位数统计 分数 15

全屏浏览 切换布局 作者 CHEN, Yue 单位 浙江大学

给定一个 k 位整数 $N = d_{k-1}10^{k-1} + \cdots + d_1 10^1 + d_0$ $(0 \le d_i \le 9, i = 0, \cdots, k-1, d_{k-1} > 0)$，请编写程序统计每种不同的个位数字出现的次数。例如：给定 $N = 100311$，则有 2 个 0，3 个 1，和 1 个 3。

输入格式：

每个输入包含 1 个测试用例，即一个不超过 1000 位的正整数 N。

输出格式：

对 N 中每一种不同的个位数字，以 `D:M` 的格式在一行中输出该位数字 `D` 及其在 N 中出现的次数 `M`。要求按 `D` 的升序输出。

输入样例：

```
100311
```

输出样例：

```
0:2
1:3
3:1
```

核心代码：

```
int main(){
    //注意a[]、b[]的数组大小
    int a[1000], b[10]={0}, k=0, num=0;
    string str;
    getline(cin, str);
    do{
        a[k] = str[k] - '0';
    }while(str[++k] != '\0');
    for(int i=0;i<k;i++){
        b[a[i]]++;
    }
    for(int i=0;i<10;i++){
        if(b[i] != 0) printf("%d:%d\n", i, b[i]);
    }
    return 0;
}
```

5、科学计数法：

科学计数法是科学家用来表示很大或很小的数字的一种方便的方法，其满足正则表达式 [+-][1-9].[0-9]+E[+-][0-9]+，即数字的整数部分只有 1 位，小数部分至少有 1 位，该数字及其指数部分的正负号即使对正数也必定明确给出。

现以科学计数法的格式给出实数 *A*，请编写程序按普通数字表示法输出 *A*，并保证所有有效位都被保留。

输入格式：

每个输入包含 1 个测试用例，即一个以科学计数法表示的实数 *A*。该数字的存储长度不超过 9999 字节，且其指数的绝对值不超过 9999。

输出格式：

对每个测试用例，在一行中按普通数字表示法输出 *A*，并保证所有有效位都被保留，包括末尾的 0。

输入样例 1：

```
+1.23400E-03
```

输出样例 1：

```
0.00123400
```

输入样例 2：

```
-1.2E+10
```

输出样例 2：

```
-12000000000
```

核心代码：

```
int main(){
    string str;
    getline(cin, str);

    int zhishu=0, i;
    char fuhao;
    string shu, xiaoshu;
    for(i=0;i<str.size();i++){
        if(i == 1) shu.push_back(str[i]);           //记录整数部分
        else if(i > 2 && str[i] != 'E')
            xiaoshu.push_back(str[i]);              //记录小数部分,不记录小数
点
        else if(str[i] == 'E'){
            fuhao = str[++i]; break;                //记录指数的符号
        }
    }
    for(int j=0;j<4 && i+1+j<str.size();j++){
        zhishu = zhishu*10 + str[i+1+j] - '0';                  //记录指数
的值
    }

    if(fuhao == '+'){                                           //指数是正
值，做乘法
        for(int j=0;j<zhishu;j++){
            if(j < xiaoshu.size()) shu.push_back(xiaoshu[j]);   //先拼凑小
数部分的数字
            else shu.push_back('0');                            //后拼凑数
字0
        }
        if(zhishu < xiaoshu.size()){                            //如果指数
小于小数部分长度
            shu.push_back('.');                                 //需要添加
小数点
            for(int j=zhishu;j<xiaoshu.size();j++){
                shu.push_back(xiaoshu[j]);                      //并将剩余
部分的小数部分是数字拼凑上去
            }
        }
    }
    else{                                                       //指数是负
值，做除法
        for(int j=0;j<xiaoshu.size();j++){                      //整合整数
与小数
            shu.push_back(xiaoshu[j]);
        }
        for(int j=0;j<zhishu;j++){
            if(j == zhishu-1) shu.insert(0, "0.");
```

```
            else shu.insert(0, "0");
        }
    }
    if(str[0] == '+') cout<<shu<<endl;
    else cout<<'-'<<shu<<endl;
    return 0;
}
```

6、查验身份证：

一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；然后将计算的和对11取模得到值`Z`；最后按照以下关系对应`Z`值与校验码`M`的值：

```
Z：0 1 2 3 4 5 6 7 8 9 10
M：1 0 X 9 8 7 6 5 4 3 2
```

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

输入格式：

输入第一行给出正整数*N*（≤100）是输入的身份证号码的个数。随后*N*行，每行给出1个18位身份证号码。

输出格式：

按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出`All passed`。

核心代码：

```
char M[12] = {'1','0','X','9','8','7','6','5','4','3','2'};
int W[17] = {7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2};
int main(){
    int N, num=0, flag=1;    //flag记录当前身份证号的前17位是否全为数字
    scanf("%d", &N);
    getchar();

    string ID[N];
    int sum[N] = {0};
    for(int i=0;i<N;i++){
        getline(cin, ID[i]);
        flag = 1;            //!!!!!!!每个循环要刷新一次符号标志

        for(int j=0;j<17;j++){
            if(ID[i][j] < '0' || ID[i][j] > '9'){    //若前17位不全为数字
                flag = 0; break;
            }
```

```
            sum[i] = sum[i] + (ID[i][j]-'0')*W[j];    //前17位加权求和
        }
        if(flag == 1){
            int z = sum[i] % 11;                      //对11取模
            if(ID[i][17] != M[z]) flag = 0;           //校验位不准确
        }

        if(flag == 0){
            num++;                                     //记录不合法的身份证号
数量
            cout<<ID[i]<<endl;
        }
    }
    if(num == 0) printf("All passed\n");
    return 0;
}
```

二、甲类题目：

1、 A+B Format

Calculate *a*+*b* and output the sum in standard format -- that is, the digits must be separated into groups of three by commas (unless there are less than four digits).

Input Specification:

Each input file contains one test case. Each case contains a pair of integers *a* and *b* where −106≤*a*,*b*≤106. The numbers are separated by a space.

Output Specification:

For each test case, you should output the sum of *a* and *b* in one line. The sum must be written in the standard format.

Sample Input:

```
-1000000 9
```

Sample Output:

```
-999,991
```

Key Codes:

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    cin >> a >> b;
    string s = to_string(a + b);
    int len = s.length();
```

```
    if(s[0] == '-'){
        cout<< '-';
        s.erase(0, 1);     //从位置 0 开始，删除 1 个字符
    }
    int count = 0;         //从低位到高位记录当前位置
    for(int i=s.length()-1;i>=0;i--){
        count++;
        if(count % 3 ==0 && i>0){     //i>0,即位置 0 之前无需添加逗号
            s.insert(i, ",");
        }
    }
    cout<<s<<endl;
    return 0;
}
```

2、Kuchiguse——最长公共后缀

The Japanese language is **notorious** for its sentence ending **particles**. Personal preference of such particles can be considered as a reflection of the speaker's personality. Such a preference is called "**Kuchiguse**" and is often exaggerated artistically in **Anime** and **Manga**. For example, the artificial sentence ending particle "nyan~" is often used as a stereotype for characters with a cat-like personality:

- Itai nyan~ (It hurts, nyan~)
- Ninjin wa iyada nyan~ (I hate carrots, nyan~)

Now given a few lines spoken by the same character, can you find her Kuchiguse?

Input Specification:

Each input file contains one test case. For each case, the first line is an integer N ($2 \le N \le 100$). Following are N file lines of **0~256 (inclusive) characters** in length, each representing a character's spoken line. The spoken lines are **case sensitive**.

Output Specification:

For each test case, print in one line the kuchiguse of the character, i.e., **the longest common suffix of all *N* lines**. If there is no such suffix, write `nai`.

Sample Input 1:

```
3
Itai nyan~
Ninjin wa iyadanyan~
uhhh nyan~
```

Sample Output 1:

```
nyan~
```

Sample Input 2:

```
3
Itai!
Ninjinnwaiyada T_T
T_T
```

Sample Output 2:

```
nai
```

Key Codes:

```
#include <iostream>
#include <algorithm>
using namespace std;
int main(){
    int n, minLen=256, ans=0;
    string s[100];
    scanf("%d", &n);
    getchar();      //接收换行符
    for(int i=0;i<n;i++){
        getline(cin, s[i]);
        int len = s[i].size();
        if(len < minLen) minLen = len;      //记录最小长度
        reverse(s[i].begin(), s[i].end());  //字符串反转
    }
    for(int i=0;i<minLen;i++){
        char c = s[0][i];                   //记录第一个字符串的第i个字符
        bool same = true;
        for(int j=1;j<n;j++){               //比较所有字符串的第i个字符
            if(c != s[j][i]) {same = false;break;}
        }
        if(same) ans++;                     //记录公共后缀的长度
        else break;
    }
    if(ans > 0){
        for(int i=ans-1;i>=0;i--){printf("%c", s[0][i]);}
    }
    else printf("nai");
    return 0;
}
```

3、Read Number in Chinese

Given an integer with no more than 9 digits, you are supposed to read it in the traditional Chinese way. Output `Fu` first if it is negative. For example, -123456789 is read as `Fu yi Yi er Qian san Bai si Shi wu Wan liu Qian qi Bai ba Shi jiu`. Note: zero (`ling`) must be handled correctly according to the Chinese tradition. For example, 100800 is `yi Shi Wan ling ba Bai`.

Input Specification:

Each input file contains one test case, which gives an integer with no more than 9 digits.

Output Specification:

For each test case, print in a line the Chinese way of reading the number. The characters are separated by a space and there must be no extra space at the end of the line.

Sample Input 1:

```
-123456789
```

Sample Output 1:

```
Fu yi Yi er Qian san Bai si Shi wu Wan liu Qian qi Bai ba Shi jiu
```

Sample Input 2:

```
100800
```

Sample Output 2:

```
yi Shi Wan ling ba Bai
```

Key Codes:

- 若在数字的某节（个节、万节、亿节）中，某个非零位的高位为零，则需额外输出一个零；
- 每节末尾要输出 Wan 或 Yi；

解决思路：

- [ ] **使用 left 表示每节的最高位，right 表示每节的个位；**使用 flag 表示是否有累积的0；使用 **isPrint** 表示该节是否输出过其中的位（从而确定该节是否全为0，若全为0，则不需要 Wan 或 Yi）；
- [ ] 循环：每次循环 left++；
  - [ ] 如果当前位为0，令 flag = true；
  - [ ] 否则：①判断flag是否为true，以确认是否需要输出 ling；
    ②输出当前位的数字，并置 isPrint = true;
    ③若 left 指向该节的最低位，判断是否输出 Wan 或 Yi；
- [ ] 上面一个循环就代表着一个节执行完毕；

```
#include <iostream>
#include <cmath>
#include <algorithm>
using namespace std;
char num[10][5] = {"ling","yi","er","san","si","wu","liu","qi","ba","jiu"};
char wei[5][5] = {"Shi", "Bai", "Qian", "Wan", "Yi"};
```

```
int main(){
    string str;
    getline(cin, str);
    int len = str.size();            //字符串长度
    int left = 0, right = len-1;     //left、right分别指向字符串首尾元素
    if(str[0] == '-'){               //若为负数，left右移一位
        printf("Fu");left++;
    }
    while(left + 4 <= right){        //将right每次左移4位，直至left、right在同一节
        right -= 4;
    }
    while(left < len){
        bool flag = false;          //用于表示是否有累积的0
        bool isPrint = false;       //表示该节是否输出过其中的位
        while(left <= right){
            if(left > 0 && str[left] == '0'){                //若当前位为0
                flag = true;
            }
            else{                                            //若当前位不为0
                if(flag){printf(" ling"); flag = false;}     //有累积的0
                if(left > 0) printf(" ");                    //不是首位数字，就要输出空格

                int zhi = str[left]-'0';
                printf("%s", num[zhi]);                      //输出当前位的数字
                isPrint = true;
                if(left != right){
                    printf(" %s", wei[right - left -1]);     //输出十、百、千
                }
            }
            left++;
        }
        if(isPrint==true && right!=len-1){                   //不是个节的个位，要输出万或亿
            printf(" %s", wei[(len-1-right)/4 + 2]);
        }
        right += 4;    //进入下一节
    }
    return 0;
}
```

附录1：ASCII码表

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	”	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	—	127	DEL