1.组装数字

> 输入两个正整数，a，b。
> a可以组装到b的左边也可以组装到b的右边。例如a = 2，b = 16 则c = 216或者c = 162。那么如果我们需要的是其中那个大的数字也就是216的话，你能编写程序，帮我们找出由a，b组装而成的大的那个数值吗?
>
> ---
>
> 输入
>
> > 输入包含两行，每个整数占一行。每个整数均为不超过10位的正整数。
>
> ---
>
> 输出
>
> > 输出一个整数，即由a，b组装而成的那个相对大的数字。

方法一：循环+比较

```c
#include <stdio.h>
int main(){
    char a[11],b[11];
    int i;
    scanf("%s%s",a,b);
    for (i=0;i<10;i++){
        if (a[i]>b[i]){
            printf("%s%s",a,b);
            break;
        }
        else if (a[i]<b[i]){
            printf("%s%s",b,a);
            break;
        }
```

```
    }
}
```

方法二：合成字符串+比较

```
#include <stdio.h>
#include <string.h>
int main() {
    char a[11], b[11];
    char num1[22], num2[22];
    scanf("%s%s", a, b);
    sprintf(num1, "%s%s", a, b);
    sprintf(num2, "%s%s", b, a);
    if (strcmp(num1, num2) > 0)  printf("%s\n", num1);
    else printf("%s\n", num2);
}
```

2.明明的随机数

> 明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数（N <= 100），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

输入

> 输入包含两行：
> 第一行为1个正整数，表示所生成的随机数的个数N。
> 第二行有N个用空格隔开的正整数，为所产生的随机数。

> 输出
>
> > 输出两行，第一行为1个正整数M，表示不相同的随机数的个数。第二行为M个用空格隔开的正整数，为从小到大排好序的不相同的随机数。

```c
#include <stdio.h>

int partition(int arr[], int low, int high) {
    int pivot = arr[high],i = (low - 1),j,temp;
    for (j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n, arr[101], i, set[101] = {0}, count = 0;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
```

```
    set[0] = arr[0];
    count = 1;
    for (i = 1; i < n; i++) {
        if (arr[i] != arr[i - 1]) {
            set[count] = arr[i];
            count++;
        }
    }
    printf("%d\n", count);
    for (i = 0; i < count - 1; i++) {
        printf("%d ", set[i]);
    }
    printf("%d\n", set[count - 1]);
}
```

3.不高兴的津津

> 津津上初中了。妈妈认为津津应该更加用功学习，所以津津除了上学之外，还要参加妈妈为她报名的各科复习班。另外每周妈妈还会送她去学习朗诵、舞蹈和钢琴。但是津津如果一天上课超过八个小时就会不高兴，而且，上得越久就会越不高兴。假设津津不会因为其它事不高兴，并且她的不高兴不会持续到第二天。请你帮忙检查一下津津下周的日程安排，看看下周她会不会不高兴；如果会的话，哪天最不高兴。

输入

> 输入包含七行，分别表示周一到周日的日程安排。
> 每行包括两个小于10的非负整数，用空格隔开，分别表示津津在学校上课的时间和妈妈安排她上课的时间。

输出

> 输出一个数字。如果不会不高兴则输出0，如果会则输出最不高兴的是周几(用1,2,3,4,5,6,7分别表示周一，周二，周三，周四，周五，周六，周日)。如果有两天或两天以上不高兴的程度相当，则输出时间最靠前的一天。

```c
#include <stdio.h>
int main(){
    int a,b,a_b=-1,c,i;
    for(i=1;i<=7;i++){
        scanf("%d %d",&a,&b);
        if(a+b>a_b){
            a_b=a+b;
            c=i;
        }
    }
    if (a_b<=8) printf("0");
    else printf("%d",c);
}
```

4.母亲节的鲜花

> 母亲节的前一天，皮皮来到鲜花店，准备去买一些花送给妈妈。花有n种，他身上却只有m元钱，每种花的价格都不一样。皮皮非常希望知道他最多可以买多少朵花。请你帮助他吧！
>
> ---
>
> 输入
>
> > 输入包含两行：
> > 第一行有两个整数n和m，用一个空格隔开。
> > 第二行有n个整数，代表每种花单枝的价格，邻近两数之间用一个空格隔开。
>
> ---

> 输出
>
> > 输出最多能够购买的花的数量。

由于没说可以重复买同一种花，因此这题有一定的误导性。

```
#include <stdio.h>

int partition(int arr[], int low, int high) {
    int pivot = arr[high],i = (low - 1),j,temp;
    for (j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main(){
    int n,m,price[1000],i;
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++){
        scanf("%d",&price[i]);
    }
    quickSort(price,0,n-1);
```

```
    printf("%d",m/price[0]);
}
```

5.平衡数对

> 若自然数x,y满足$\frac{1}{x}+\frac{1}{y}=\frac{1}{n}$，则称数对(x,y)是n平衡数对。
> 给出自然数n，求n平衡数对的数量。
>
> ---
>
> 输入
>
> > 输入一个整数n。
>
> ---
>
> 输出
>
> > 输出一个整数，表示n平衡数对的数量。

$$\frac{1}{x}+\frac{1}{y}=\frac{1}{n}$$

$$xy=n(x+y)$$

$$(x-n)(y-n)=n^2$$

记$a=x-n$，$b=y-n$，则$x=a+n$，$y=b+n$
由于$(1,n^2)$和$(n^2,1)$是不同的两种解，因此题目的结果等于n^2的正因数数目。

代码原理：

1. **质因数分解**：遍历 22 到 nn 的数，找出所有质因数及其指数。
2. **累乘计算**：对每个质因数 p_i，将指数 e_i 代入公式 $2e_i+1$，并将结果累乘。

3. **处理剩余质因数**：若分解后 n>1，说明存在最后一个质因数（指数为 1），此时乘 3（即 $2 \times 1 + 1$）

```c
#include <stdio.h>

int main() {
    int n,i,result=1;
    scanf("%d", &n);
    for (i = 2; i * i <= n; ++i) {
        if (n % i == 0) {
            int exponent = 0;
            while (n % i == 0) {
                exponent++;
                n /= i;
            }
            result *= 2 * exponent + 1;
        }
    }
    if (n > 1) {
        result *= 3;
    }
    printf("%d", result);
}
```

6.较大的数

> 在数列中有这样一种数，它们不在数列中的第一个，也不在最后一个，而且刚好都比左边和右边相邻的数都大。而这类数就叫做“较大的数”。请你按数列的顺序逐个输出所有“较大的数”。
>
> ---
>
> 输入

> 输入包含两行：
> 第一行是n（3 <= n <= 1000）。
> 第二行是n个整数，邻近两数之间用一个空格隔开。

输出

> 输出数列中所有“较大的数”，邻近两数之间用一个空格隔开。

```c
#include <stdio.h>

int main(){
    int n,arr[1000],i;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(i=1;i<n-1;i++){
        if(arr[i]>arr[i-1] && arr[i]>arr[i+1]){
            printf("%d ",arr[i]);
        }
    }
}
```

7.单词接龙

> 单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”(每个单词都最多在“龙”中出现两次)，在两个单词相连时，其重合部分合为一部分，例如beast和astonish，如果接成一条龙则变为 `beastonish`，另外相邻的两部分不能存在包含关系，例如at和 `atide` 间不能相连。

> 输入
>
> > 输入包含n + 2行：
> > 第一行为一个单独的整数n（1 <= n <= 10）表示单词数
> > 第二行到第n + 1行每行有一个单词字符串（1 <= 字符串长度 <= 99）。
> > 第n + 2行为一个单个字符，表示“龙”开头的字母。可以假定以此字母开头的“龙”一定存在。
>
> ---
>
> 输出
>
> > 只需输出以此字母开头的最长的“龙”的长度。

这显然是一个`dfs`(深度优先)算法，但是本题时间复杂度很高，约为$O(n! \times 2^n \times L)$，

```c
#include <stdio.h>
#include <string.h>

#define MAX_WORDS 20
#define MAX_WORD_LEN 100
#define MAX_CURRENT_LEN 100000

int n;
char words[MAX_WORDS][MAX_WORD_LEN];
int used[MAX_WORDS] = {0};
char start[2];
char current_str[MAX_CURRENT_LEN];
int current_len;
int max_length;

int canConnect(const char* word1, int len1, const char*
word2) {
    int len2 = strlen(word2);
    int min_len = len1 < len2 ? len1 : len2;
    int i;
```

```c
    for (i = 1; i < min_len; ++i) {
        if (strncmp(word1 + len1 - i, word2, i) == 0) {
            return i;
        }
    }
    return 0;
}

void dfs() {
    int i;
    if (current_len > max_length) {
        max_length = current_len;
    }
    for (i = 0; i < n; ++i) {
        if (used[i] < 2) {
            int overlap = canConnect(current_str,
current_len, words[i]);
            if (overlap > 0) {
                int len_i = strlen(words[i]);
                int append_len = len_i - overlap;
                int prev_len = current_len;
                strcpy(current_str + prev_len, words[i]
+ overlap);
                current_len += append_len;
                used[i]++;
                dfs();
                used[i]--;
                current_len = prev_len;
                current_str[prev_len] = '\0';
            }
        }
    }
}

int main() {
    int i;
    scanf("%d", &n);
    for (i = 0; i < n; ++i) {
        scanf("%s", words[i]);
    }
```

```
    scanf("%s", start);

    max_length = 0;
    for (i = 0; i < n; ++i) {
        if (words[i][0] == start[0]) {
            used[i]++;
            strcpy(current_str, words[i]);
            current_len = strlen(words[i]);
            dfs();
            used[i]--;
        }
    }
    printf("%d\n", max_length);
}
```

8.猪八戒办宴会

> 西天取经已经过去1000年了，猪八戒很想念师兄弟们。他给师兄弟、师傅还有那匹白马都发了邀请信，请他们来猪八戒的高老庄赴宴。对了，还邀请了月宫的诸位仙女们。有时间赴宴的客人共n（1 <= n <= 100）位，他们都给猪八戒回信，而且在信中特别强调了自己最喜欢的一种水果，要求猪八戒把这种水果作为他/她的餐前水果。猪八戒受到回信后就赶紧准备起来，去世界各地采购这些水果，好在客人们最喜欢的水果各不相同。
> 经过一个月的精心准备，宴会马上要开始了，已经把水果按照客人喜好摆在每位客人的餐桌上了。有些客人的水果数量要多些，比如猴哥；有些客人的水果数量要少些，比如沙和尚。根据管家采购的水果单子，请你判断客人数量。
>
> ---
>
> 输入
>
> > 输入采购的水果单子列表。每种水果必定为一个全小写的英文单词（不含空格）。水果单子字符串用回车结束，水果单子字符串长度不

> 超过1999个字符，水果名称不超过20个字符。每两个邻近的英文单词之间用一个空格隔开。请注意的是，水果单子中，同种水果出现的次数表示该种水果的采购数量。

输出

> 输出客人数量n，n是正整数。

这题的叙述其实不是很准确，样例给的也不是很有特性，其实这道题就是问你有多少种水果而已

```c
#include <stdio.h>
#include <string.h>

int main() {
    int i, j, count = 0;
    char fruits[1000][20];
    int n = 0;

    while (scanf("%s", fruits[n]) == 1) {
        n++;
    }

    for (i = 0; i < n; i++) {
        int is_new = 1;
        for (j = 0; j < i; j++) {
            if (strcmp(fruits[i], fruits[j]) == 0) {
                is_new = 0;
                break;
            }
        }
        if (is_new) {
            count++;
        }
    }
```

```
    printf("%d", count);
}
```

9.宝宝读文件左移数列

> 已知文本文件file.txt中连续存放了n（2 <= n <= 1000）个整数，邻近两数之间用一个空格隔开。第一个数为文件中存放整数的总数n，后面n - 1个数为具体的数值，这些数有正有负，也没有被排序。
> 编写程序，读入文件中的数，并且将后面n - 1个数读入数组中。由于宝宝讨厌正数，想把这些数向左平移，移动规则是让数列中的最大值落在坐标轴原点上。宝宝的方法是先找出数列中的最大值，然后将数列中的每个数都减去最大值，这样就实现数列向左平移了。
> 注意：1.路径及文件名为'*file.txt*'，注意不要写路径。2.只允许使用只读方式

输入

> 读文件输入。

输出

> 输出减去最大值之后的n - 1个整数，邻近两数之间用一个空格隔开。

```
#include <stdio.h>

int main() {
    FILE *fp;
    int n, i, arr[1000], max;
    fp = fopen("file.txt", "r");
    if (fp == NULL) {
        return 1;
    }
```

```
    fscanf(fp, "%d", &n);
    for (i = 0; i < n - 1; i++) {
        fscanf(fp, "%d", &arr[i]);
    }
    fclose(fp);

    max = arr[0];
    for (i = 1; i < n - 1; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    for (i = 0; i < n - 1; i++) {
        arr[i] -= max;
    }

    for (i = 0; i < n - 1; i++) {
        if (i != n - 2) {
            printf("%d ", arr[i]);
        } else {
            printf("%d", arr[i]);
        }
    }
}
```

10.读文件寻找最小值

> 已知文本文件 `file4.txt` 中连续存放了20个整数，邻近两数之间用一个空格隔开。
> 编写程序，读入文件中的整数，找出其中的最小值并输出。
> 注意：1.路径及文件名为"file4.txt"，注意不要写路径。2.只允许使用只读方式打开文件。

> 输入
>
> > 读文件输入。
>
> ---
>
> 输出
>
> > 输出最小值。

```c
#include <stdio.h>

int main() {
    FILE *fp;
    int num, min;
    int i;

    fp = fopen("file4.txt", "r");
    if (fp == NULL) {
        return 1;
    }

    fscanf(fp, "%d", &min);
    for (i = 1; i < 20; i++) {
        fscanf(fp, "%d", &num);
        if (num < min) {
            min = num;
        }
    }

    fclose(fp);
    printf("%d", min);
}
```