1.苹果达人

> 围坐成一圈的12个小朋友，每个人身上都有若干个苹果，老师要选出哪5个挨着坐的小朋友手中的苹果总数最多，你能帮助老师们编写一个程序来确定吗？请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入12个整数，表示12个小朋友分别的苹果数。邻近两数之间用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出从第几个小朋友开始苹果总数最多。

方法一（最优解）：这是经典的动态规划——打家劫舍的问题

```c
#include <stdio.h>
int main() {
    int lst[12], i;
    int current_sum = 0, max_sum = 0;
    int count = 0;
    for (i = 0; i < 12; i++) {
        scanf("%d", &lst[i]);
        if (i < 5) {
            current_sum += lst[i];
        }
    }

    max_sum = current_sum;
    count = 0;

    for (i = 1; i < 12; i++) {
        current_sum = current_sum - lst[i - 1] + lst[(i + 4) % 12];

        if (current_sum > max_sum || (current_sum == max_sum && i < count))
{
            max_sum = current_sum;
            count = i;
        }
    }
```

```
    printf("%d\n", count + 1);
}
```

方法二：数组+求和后的数组比较

```
#include <stdio.h>
int main() {
    int lst[12], i,m,n,p,q;
    int sum[12],max_sum;
    int count = 0;
    for (i = 0; i < 12; i++) {
        scanf("%d", &lst[i]);
    }


    for (i = 0; i < 12; i++) {
        m = (i+1)%12;
        n = (i+2)%12;
        p = (i+3)%12;
        q = (i+4)%12;
        sum[i] = lst[i]+lst[m]+lst[n]+lst[p]+lst[q];
    }
    max_sum = sum[0];
    count = 0;
    for (i = 1; i < 12; i++) {
        if(sum[i] > max_sum) {
            max_sum = sum[i];
            count = i;
        }
    }

    printf("%d\n", count + 1);
}
```

2.陶陶摘苹果

> 陶陶家的院子里有一棵苹果树，每到秋天树上就会结出10个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个30厘米高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。
> 现在已知10个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。请使用循环和数组实现。

输入

> 输入包含两行：
> 第一行是10个100到200之间（包括100和200）的整数（以厘米为单位）。分别表示10个苹果到地面的高度，两个相邻的整数之间用一个空格隔开。
> 第二行是一个100到120之间（包含100和120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度。

输出

> 输出一个整数，表示陶陶能够摘到的苹果的数目。

```c
#include <stdio.h>
int main(){
    int lst[10],n,i,count=0;
    for(i=0;i<10;i++) {
        scanf("%d", &lst[i]);
        lst[i] -= 30;
    }
    scanf("%d", &n);
    for(i=0;i<10;i++) {
        if(lst[i]<=n) {
            count++;
        }
    }
    printf("%d", count);
}
```

3.统计单词的平均长度

> 输入若干个单词，输出它们的平均长度，保留两位小数。单词只包含大写字母和小写字母，单词前后都可能有一个或者多个空格隔开。请使用循环和数组实现。

输入

> 输入若干个单词，单词前后都可能有一个或者多个空格隔开。

输出

> > 输出平均长度，小数点后必须保留2位有效数字（四舍五入），不足补零。

为了防止这一行输出以空格（\t）结尾导致程序不能正常运行，使用ungetc保证不干扰str录入的情况下，保证可以正常运行。
当然，这题在站内测试运行的时候，好像也不会出现这种情况。

```
#include <stdio.h>
#include <string.h>
int main(){
    char str[1000][100],m;
    int i,len=0,count=0;
    float n;
    for(i=0;i<100;i++) {
        scanf("%s%c", str[i], &m);
        len += strlen(str[i]);
        count++;
        if (m == ' ') scanf("%c", &m);
        if (m == '\n') break;
        else ungetc(m, stdin);
    }
    n = (float)len/count;
    printf("%.2f",n);
}
```

4.喝醉的狱卒

> 在一所监狱里有一条长长的走廊，沿着走廊排列着n个牢房，编号为1到n。每个牢房有一个囚犯，而且牢房的门都是锁着的。
> 一天晚上，狱卒很无聊，于是他就玩起了一个人的游戏。第一轮，他喝了一口威士忌，然后沿着走廊，将所有牢房的门打开。第二轮，他又喝了一口威士忌，然后又沿着走廊，将所有编号为2的倍数的牢房锁上。第三轮，他再喝一口威士忌，再沿着走廊，视察所有编号为3的倍数的牢房，如果牢房是锁着的，他就把它打开；如果牢房是开着的，他就把它锁上。他如此玩了n轮后，喝下最后一口威士忌，醉倒了。
> 当他醉倒后，一些犯人发现他们的牢房开着而且狱卒已经无能为力，他们立刻逃跑了。
> 给出牢房的数目n，请你确认最多有可能有多少犯人逃出了监狱？请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入一个正整数n（0 < n < 1000）。表示狱卒喝了多少轮酒。
>
> ---

> 输出
>
> > 输出一个正整数m，表示能够逃跑的人数。

方法一：老老实实用数组

```
#include <stdio.h>
int main(){
    int n,m=0,i,j;
    int sign[1000];
    scanf("%d",&n);
    for(i=1;i<=n;i++){
        sign[i-1] = 1;
        for(j=1;j<=i;j++){
            if(i%j==0){
                sign[i-1] = sign[i-1] * -1;
            }
        }
    }
    for(i=0;i<n;i++){
        if(sign[i]<0){
            m++;
        }
    }
    printf("%d",m);
}
```

方法二：不用数组（代码验证不一定过，所以我耍了点小聪明，将n换成n[1]数组

```
#include <stdio.h>
int main(){
    int n[1],m=0,i,j;
    scanf("%d",&n[0]);
    for(i=1;i<=n[0];i++){
        int sign= 1;
        for(j=1;j<=i;j++){
            if(i%j==0){
                sign = sign * -1;
        }
    }
        if(sign<0){
            m++;
        }
    }
    printf("%d",m);
}
```

方法三：其实这题并没有这么麻烦，换个角度想，***只有牢房号的因数为奇数的人才能逃出去 → 只有平方数的犯人可以跑出去***
当然，我不保证代码验证能过

```
#include <stdio.h>
#include <math.h>
int main(){
    int n,m;
    scanf("%d",&n);
    m = (int)sqrt(n);
    printf("%d",m);
}
```

5.奇数还是偶数

> 那么很简单，给你一个二进制数字。请你帮我判定这个数字是奇数还是偶数？请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入一个不会超过50位的二进制正整数。
>
> ---
>
> 输出
>
> > 如果是奇数则输出“ODD”，如果是偶数则输出“EVEN”（不输出引号）。

方法一：老老实实用数组

```
#include <stdio.h>
int main() {
    int  i = 0;
    char c,lst[50][1],sign;
    while (1) {
        scanf("%c", &c);
        if (c == '\n'||c==' ') {
            break;
        } else ungetc(c, stdin);
        scanf("%c", lst[i]);
        if (lst[i][0]=='0') sign = '0';
        else sign = '1';
        i++;
```

```
    }
    if (sign=='0') printf("EVEN");
    else printf("ODD");
}
```

方法二：不用数组，同样，不保证代码验证能过

```
#include <stdio.h>
int main(){
    int n[1],m;
    scanf("%d",&n[0]);
    while(0) break;
    m = n[0] % 10;
    if (m%2==0) printf("EVEN");
    else printf("ODD");
}
```

6.分数统计

> 输入N个学生的英语分数，哪个分数出现的次数最多？如果有多个并列，从小到大输出。分数均为不超过100的非负整数。请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入包含两行：
> > 第一行是一个整数N（0 < N <= 300）。
> > 第二行是N个非负整数，分别表示N个学生分数，两个相邻的整数之间用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出出现次数最多的分数。如有多个，用空格隔开，并从小到大输出。

```
#include <stdio.h>
int main() {
    int N,i;
    int grade[300],counts[101] = {0},max_count = 0,first_print = 1;
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &grade[i]);
```

```
            counts[grade[i]]++;
        }
        for (i = 0; i <= 100; i++) {
            if (counts[i] > max_count) {
                max_count = counts[i];
            }
        }
        for (i = 0; i <= 100; i++) {
            if (counts[i] == max_count) {
                if (first_print) {
                    printf("%d", i);
                    first_print = 0;
                } else {
                    printf(" %d", i);
                }
            }
        }
    }
```

7.排序

> 小明得到一个任务，要对n个整数进行排序。他向你求援，希望你能够帮助他，将输入的n个数值按照从小到大的顺序排序后输出。请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入包含两行：
> > 第一行包括一个整数n，表示n个整数（1 < n <= 300）。
> > 第二行为n个整数，邻近两数之间用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出从小到大的顺序排序后的数列，相邻的数之间用一个空格隔开。

方法一：穷举法（原始冒泡排序）
时间复杂度$O(n^2)$

```
#include <stdio.h>
void fun(int arr[], int t) {
    int i,j;
    for (i = 0; i < t-1; i++) {
```

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

方法二：双指针法（鸡尾酒排序）
时间复杂度：最好$O(n)$，最坏$O(n^2)$，接近$O(n^2)$

```
#include <stdio.h>
void fun(int arr[], int t) {
    int left = 0,right = t - 1,j;
    while (left < right) {
        for (j = left; j < right; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
        right--;
        for (j = right; j > left; j--) {
            if (arr[j - 1] > arr[j]) {
                int temp = arr[j];
                arr[j] = arr[j - 1];
                arr[j - 1] = temp;
            }
        }
        left++;
    }
}
int main() {
```

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

方法三：快速排序
时间复杂度：$O(n\log n)$，最坏$O(n^2)$，但是更接近于$O(n\log n)$

```
#include <stdio.h>
void quick_sort(int arr[], int left, int right) {
    int pivot = arr[(left + right) / 2],i = left, j = right;
    if (left >= right) return;
    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;
        if (i <= j) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
    }
    quick_sort(arr, left, j);
    quick_sort(arr, i, right);
}
void fun(int arr[], int t) {
    quick_sort(arr, 0, t - 1);
}
int main() {
    int N, arr[1000], i;
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }
    fun(arr, N);
    for (i = 0; i < N; i++) {
        printf("%d", arr[i]);
        if (i != N-1) printf(" ");
    }
```

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

方法四：堆排序
时间复杂度：$O(n \log n)$

```
#include <stdio.h>
void heapify(int arr[], int n, int i) {
    int smallest = i,left = 2 * i + 1,right = 2 * i + 2;
    if (left < n && arr[left] > arr[smallest]) {
        smallest = left;
    }
    if (right < n && arr[right] > arr[smallest]) {
        smallest = right;
    }
    if (smallest != i) {
        int temp = arr[i];
        arr[i] = arr[smallest];
        arr[smallest] = temp;
        heapify(arr, n, smallest);
    }
}

void heap_sort(int arr[], int n) {
    int i;
    for (i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    for (i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}

void fun(int arr[], int t) {
    heap_sort(arr, t);
}

int main() {
    int N, arr[1000], i;
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }
    fun(arr, N);
    for (i = 0; i < N; i++) {
```

```
        printf("%d", arr[i]);
        if (i != N-1) printf(" ");
    }
    printf("\n");
}
```

8.允许并列的排名

> 在我们参加的各种竞赛中，允许并列的排名方式是经常遇到的。例如有四名选手的成绩分别为50、80、50、30分，则80分的选手为第一名，50分的两名选手均为第二名，30分的选手为第三名。
> 请编写一个程序，计算一个选手在这种排名方式之下的名次（分数高的选手排前面）。请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入包含三行：
> > 第一行为一个整数N，表示参赛的选手数，1 <= N <= 100。
> > 第二行为N个整数，表示每位选手的成绩（邻近两数之间用一个空格隔开）。
> > 第三行为一个整数m，表示要查询名次的选手的成绩。
>
> ---
>
> 输出
>
> > 输出一个整数，表示该选手的名次。

```
#include <stdio.h>
int main() {
    int N,arr[100]={-1},m,temp,i,j,times,site=1;
    scanf("%d",&N);
    for (i=0;i<N;i++){
        times=0;
        scanf("%d",&temp);
        for (j=0;j<i;j++){
            if(arr[j]==temp){
                times++;
                }
            }
        if(times==0){
            arr[i]=temp;
        }
    }
```

```
    scanf("%d",&m);
    for (i=0;i<N;i++){
        if(arr[i]>m){
            site++;
            }
        }
    printf("%d",site);
}
```

9.13号星期几

> 请编程统计：从1900年1月1日（当天是星期一）开始经过的n年当中，每个月的13号这一天是星期一、星期二、星期三、……、星期日的次数分别是多少？请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入一个整数n (1 <= n <= 100)。
>
> ---
>
> 输出
>
> > 输出7个整数（依次是星期一、星期二、星期三、……、星期日的次数），各数间以空格相隔。

```
#include <stdio.h>
int isLeapYear(int year) {
    if (year % 4 != 0) return 0;
    if (year % 100 != 0) return 1;
    return (year % 400 == 0);
}
int main() {
    int n,i,year,month;
    int weekCount[7] = {0};
    int currentDay = 0;
    int monthDays[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    scanf("%d", &n);
    for (year = 1900; year < 1900 + n; year++) {
        for (month = 0; month < 12; month++) {
            int day13 = (currentDay + 12) % 7;
            int daysInMonth = monthDays[month];
            weekCount[day13]++;
```

```
            if (month == 1 && isLeapYear(year)) {
                daysInMonth = 29;
            }
            currentDay = (currentDay + daysInMonth) % 7;
        }
    }
    for (i = 0; i < 7; i++) {
        printf("%d", weekCount[i]);
        if (i < 6) printf(" ");
    }
    printf("\n");
}
```

10.贪心的武松

> 打虎英雄武松接到了景阳岗动物园的求助信，信上说：最近我们动物园逃跑了N只老虎，请您把它们抓回来！武松接到信后立刻赶到了景阳岗。当他走到半山腰时，发现了这群老虎。每只老虎脖子上都挂着一块牌子，上面写着它的武力值（整数）。武松提出与老虎逐个单挑，输了的老虎就要回到动物园。老虎们同意了。
> 打虎是个体力活，当武松与某只老虎PK时，只有武松的武力值M不小于这只老虎的武力值Ai，武松才能赢！当然武松此时的体力值也将减去Ai，而且不能恢复。
> 武松想多抓几只老虎回去，请你帮他算一算，武松最多能抓几只老虎回动物园？（提示：老虎们很讲究绅士风度，不会群殴，而是一只一只上的！）请使用循环和数组实现。
>
> ---
>
> 输入
>
> > 输入包含3行：
> > 第一行包括一个整数M，表示武松的武力值。
> > 第二行包括一个整数N，表示N只老虎（1 < N < 100）。
> > 第三行为N个整数，表示每只老虎的武力值，邻近两数之间用一个空格隔开。
>
> ---
>
> 输出
>
> > 输出一个整数，为武松最多能抓的老虎数。

其实这就是从小到大选，看能取多少个。所以在作业5中从大到小排序代码基础上，修改main部分代码即可。

```
#include <stdio.h>
void heapify(int arr[], int n, int i) {
```

```c
    int smallest = i,left = 2 * i + 1,right = 2 * i + 2;
    if (left < n && arr[left] > arr[smallest]) {
        smallest = left;
    }
    if (right < n && arr[right] > arr[smallest]) {
        smallest = right;
    }
    if (smallest != i) {
        int temp = arr[i];
        arr[i] = arr[smallest];
        arr[smallest] = temp;
        heapify(arr, n, smallest);
    }
}

void heap_sort(int arr[], int n) {
    int i;
    for (i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    for (i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}

void fun(int arr[], int t) {
    heap_sort(arr, t);
}

int main() {
    int N, arr[1000], i,M,count=0;
    scanf("%d", &M);
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }
    fun(arr, N);
    for (i = 0; i < N; i++) {
        if(M>=arr[i]) {
            M-=arr[i];
            count++;
        }
        else break;
    }
    printf("%d", count);
}
```