1001 printf

《论为了出不重样的签到题，能给printf玩出什么花来》

什么?你还辛辛苦苦地把上次周练的代码拿出来改（doge）

```
#include<stdio.h>
int main(){
    int a;
    while(~scanf("%d",&a))
        printf("%x\n",a);
}

```

1002 送钥匙

这题数据比较小，不知道会不会出现一些神奇的做法。这里给出两种做法。

做法一：dp

```
#include<stdio.h>
#include<string.h>//memset函数在这个头文件里
const int N=50;
int Map[N][N],dp[N][N];
void solve(){
    for(int i=1;i<=5;i++){
        for(int j=1;j<=5;j++){
            scanf("%d",&Map[i][j]);
        }
    }
    memset(dp,0,sizeof(dp));//将dp数组置零,比循环赋值为0要快
    dp[1][1]=1;
    for(int i=1;i<=5;i++){
        for(int j=1;j<=5;j++){
            if(Map[i][j])dp[i][j]+=dp[i-1][j]+dp[i][j-1];//从起点到（i,j）的路线数=从起点到(i-1,j)的路线数+从起点到(i,j-1)的路线数
        }
    }
    printf("%d\n",dp[5][5]);
}
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        solve();
    }
}
```

做法二：递归

```
#include<stdio.h>
const int N=50;
int Map[N][N];
int dfs(int x,int y){//从(x,y)到终点的路线数
    if(x==5&&y==5)return 1;//到达终点
    if(x>5||y>5)return 0;//出界
    int ans=0;
    if(Map[x][y])ans=dfs(x+1,y)+dfs(x,y+1);//从(x,y)到终点的路线数=从(x+1,y)到终点的路线数+从(x,y+1)到终点的路线数
    return ans;
}
void solve(){
    for(int i=1;i<=5;i++)
        for(int j=1;j<=5;j++)
            scanf("%d",&Map[i][j]);
    printf("%d\n",dfs(1,1));
}
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        solve();
    }
}
```

这份代码运行时会有大量重复运算，优化不难，只要开一个数组去保存已经计算过的dfs(x,y)即可让运算效率发生数量级上的变化。

1003 w学长的生日

根据题目大意，我们发现我们可以枚举x学长可以分到的蛋糕的重量ans，

来判断是不是每个人都可以和x学长一样可以分到重量ans的蛋糕

判断

$\sum_{i=0}^{n} \frac{a_i}{ans}$ >=m

如果大于等于m则枚举的ans是可以分到的，且可分到的最大的重量一定大于等于ans

否则ans不可以分到，且可分到的最大的重量一定小于等于ans。

显然ans满足二分的特性，通过二分来快速找到最大的可行解。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e5+7;

ll n,m;
ll nums[N];

bool check(ll mid){
    if(mid==0) return true;
```

```
    ll ans=0;
    for(int i=0;i<n;i++){
        ans+=nums[i]/mid;
    }
    if(ans>=m) return true;
    else return false;
}

int main(){
    while(scanf("%d%d",&n,&m)!=EOF){
        for(int i=0;i<n;i++) scanf("%d",&nums[i]);
        ll l=0,r=1e9;
        while(l<=r){
            ll mid = (l+r)/2;
            if(check(mid)) l=mid+1;
            else r=mid-1;
        }
        printf("%d\n",r);
    }
    return 0;
}
```

1004 我不想早八

思路：

这种题目是蓝桥杯最喜欢出的简单题了，算时间，年月日啥的，这个有很多做法，你可以算出起床后刷牙洗脸，买早餐后多少点，几时几分几秒，然后和八点比较，也可以**用00:00:00当做起始点，到目前时间是多少秒，把时间都转化成秒来计算**，同一单位就更好比较大小。

```
#include<stdio.h>
int main(){
    int h,m,s;
    while(~scanf("%d:%d:%d", &h, &m, &s)){
        int tmp = 8*3600;
        int need1 = h * 3600 + m * 60 + s + 17 * 60 + 47 +33;//表示是否迟到
        int need2 = h * 3600 + m * 60 + s + 24 * 60 + 47 + 12 + 33;//表示能否吃到早餐
        if( need1 < tmp ) printf("haoye\n");
        else printf("haofan\n");
        if( need2 < tmp ) printf("haoye haoye\n");
        else printf("haoe~\n");
    }
    return 0;
}
```

1005 我要拆拆拆

思路一：

一个数很显然位数越长值越大，要使和最大，所以我们尽可能的分出一个位数长的且更大。

- 情况一：分成前面若干位和最后一位
- 情况二：分成前面一位和后面若干位

两种情况取一个最大值。

思路二：

我们直接遍历每一个分割的位置，所有答案取一个最大值。

代码：

```
#include<stdio.h>
int main(){
    long long n;
    while(~scanf("%lld",&n)){
        long long t1,t2,Max = -1,mark = 10;
        while (n >= mark) {
            t1 = n % mark;
            t2 = n / mark;
            mark *= 10;
            if ( (t1 + t2) > Max ) Max = t1 + t2;
        }
        printf("%lld\n", Max);
    }
    return 0;
}
```

1006 Counting

思路：

手动模拟题意就可以发现，第n天的0、1个数和第n-1的天的0、1个数有关系，写出这个递推关系：

$$sum_1[n] = sum_1[n-1];$$

$$sum_0[n] = sum_1[n-1] + sum_0[n-1];$$

代码：

```
#include<stdio.h>
#include<string.h>
long long a[100010],b[100010];
int main(){
    int T;
    scanf("%d",&T);
    a[1] = b[1] = 1;
    for (int i = 2; i <= 50; ++i) {
        a[i] = b[i - 1];
        b[i] = a[i - 1] + b[i - 1];
    }
    while (T--) {
        int n;
        scanf("%d",&n);
        printf("%lld %lld\n",a[n],b[n]);
    }
    return 0;
}
```

1007 再来一道水题

这里给大家提一个小建议：不要把所有东西全塞到主函数里，因为这样容易导致自己看不懂自己的代码。

有人做的时候可能会担心这样暴力的做法会超时，建议参考群文件《算法竞赛从入门到进阶》p17~23学一下时间复杂度。

```
#include<stdio.h>
int n,k;
bool check(int n){
    int cou=0;
    while(n){
        if(n%10==3)++cou;
        n/=10;
    }//逐位计算有几个3
    if(cou==k)return 1;
    else return 0;
}
int main(){
    while(~scanf("%d %d",&n,&k)){
        for(;;n++){
            if(check(n)){
                printf("%d\n",n);
                break;
            }
        }
    }
}
```

1009 w学长的字符串 easy

数据范围很小，根据题目大意，模拟即可通过。

```
#include <bits/stdc++.h>
using namespace std;

const int mod=1e9+7;
const int N=1e5+7;

int n,l,x,m;
char c[N];

int main() {
    while(~scanf("%d",&l)) {
        getchar();
        for(int i=1; i<=l; i++) scanf("%c",&c[i]);
        scanf("%d",&n);
        while(n--) {
            scanf("%d",&x);
            for(int i=x,j=l-x+1;i<=j; i++,j--) {
                char t = c[i];
                c[i] = c[j];
                c[j] = t;
            }
        }
        for(int i=1;i<=l;i++) printf("%c",c[i]);
        puts("");
    }
}
```

1008 w学长的字符串hard

数据范围加强，暴力模拟肯定会超时。

首先我们举个例子：

假设 字符串 abcdef，如果我们要把1~6翻转，结果是 fedcba

可以发现，他可以分成3个操作，a和f调换位置，b和e调换位置，c和d调换位置。

其实我们可以先记录每个字符需要调换位置的次数的奇偶性，来判断是否调换位置

比如：第二个例子

abcdef

第一次输入 1，a和f调换一次，b和e调换一次，c和d调换一次。

第二次输入2，b和e调换一次，c和d调换一次。

第三次输入3，c和d调换一次。

总的来看，a和f调换一次(奇数调换)，b和e调换两次(偶数不动)，c和d调换3次(奇数调换)

结果就是fbdcea

```cpp
#include <bits/stdc++.h>
using namespace std;

const int mod=1e9+7;
const int N=1e5+7;
int n,l,x,nn;
int c[N];
int f[N];

int main() {
    while(~scanf("%d",&l)) {
        memset(f,0,sizeof(f));    //把f数组全部置为0
        memset(c,0,sizeof(c));     //把c数组全部置为0
        getchar();
        for(int i=1; i<=l; i++) scanf("%c",&c[i]);
        scanf("%d",&n);
        for(int i=0; i<n; i++) {
            scanf("%d",&x);      //统计每个位置出现了多少次
            f[x]++;
        }
        for(int i=1; i<=l/2; i++) {
            f[i]+=f[i-1];
        }
        for(int i=1; i<=l/2; i++) {
            if(f[i]%2!=0) swap(c[i],c[l-i+1]);
        }
        for(int i=1; i<=l; i++) printf("%c",c[i]);
        puts("");
    }
}
```

由上面的分析可以发现，我们只需要判断每一个位置需要交换多少次的次数是奇数还是偶数，利用异或的原理，相同则为0，不同则为1，所以我们可以利用异或的原理来判断，0 ^ 1=1,1^1=0,如此循环，循环节是2，所以你可以通过这种方式来判断奇偶

代码：

```
#include <bits/stdc++.h>
using namespace std;

const int mod=1e9+7;
const int N=1e5+7;
int n,l,x,nn;
int c[N];
int f[N];

int main() {
    while(~scanf("%d",&l)) {
        memset(f,0,sizeof(f));   //把f数组全部置为0
        memset(c,0,sizeof(c));    //把c数组全部置为0
        getchar();
        for(int i=1; i<=l; i++) scanf("%c",&c[i]);
        scanf("%d",&n);
        for(int i=0; i<n; i++) {
            scanf("%d",&x);
            f[x]^=1;//0^1=1,1^1=0,如此循环，循环节是2，所以你可以通过这种方式来判断奇偶
        }
        bool ok=0;
        for(int i=1; i<=l/2; i++) {
            if(f[i]) ok^=1;
            if(ok) swap(c[i],c[l-i+1]);
        }
        for(int i=1; i<=l; i++) printf("%c",c[i]);
        puts("");
    }
}
```