

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)

```
1 #include<stdio.h>
2 int main(){
3     int a;
4     while(~scanf("%d",&a))
5         printf("%x\n",a);
6 }
7
```

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

```
1 #include<stdio.h>
2 #include<string.h>//memset函数在这个头文件里
3 const int N=50;
4 int Map[N][N],dp[N][N];
5 void solve(){
6     for(int i=1;i<=5;i++){
7         for(int j=1;j<=5;j++){
8             scanf("%d",&Map[i][j]);
9         }
10    }
11    memset(dp,0,sizeof(dp));//将dp数组置零,比循环赋值为0要快
12    dp[1][1]=1;
13    for(int i=1;i<=5;i++){
14        for(int j=1;j<=5;j++){
15            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
16        }
17    }
18    printf("%d\n",dp[5][5]);
19 }
20 int main(){
21     int t;
22     scanf("%d",&t);
23     while(t--){
24         solve();
25     }
26 }
```

做法二：递归

```
1  #include<stdio.h>
2  const int N=50;
3  int Map[N][N];
4  int dfs(int x,int y){//从(x,y)到终点的路线数
5      if(x==5&&y==5)return 1;//到达终点
6      if(x>5||y>5)return 0;//出界
7      int ans=0;
8      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
9      return ans;
10 }
11 void solve(){
12     for(int i=1;i<=5;i++)
13         for(int j=1;j<=5;j++)
14             scanf("%d",&Map[i][j]);
15     printf("%d\n",dfs(1,1));
16 }
17 int main(){
18     int t;
19     scanf("%d",&t);
20     while(t--){
21         solve();
22     }
23 }
```

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

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

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。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N=1e5+7;
5
6  ll n,m;
7  ll nums[N];
8
9  bool check(ll mid){
10     if(mid==0) return true;
```

```

11     ll ans=0;
12     for(int i=0;i<n;i++){
13         ans+=nums[i]/mid;
14     }
15     if(ans>=m) return true;
16     else return false;
17 }
18
19 int main(){
20     while(scanf("%d%d",&n,&m)!=EOF){
21         for(int i=0;i<n;i++) scanf("%d",&nums[i]);
22         ll l=0,r=1e9;
23         while(l<=r){
24             ll mid = (l+r)/2;
25             if(check(mid)) l=mid+1;
26             else r=mid-1;
27         }
28         printf("%d\n",r);
29     }
30     return 0;
31 }

```

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，到目前为止时间是多少秒，把时间都转化成秒来计算，同一单位就更好比较大小。

```

1  #include<stdio.h>
2  int main(){
3      int h,m,s;
4      while(~scanf("%d:%d:%d",&h,&m,&s)){
5          int tmp = 8*3600;
6          int need1 = h * 3600 + m * 60 + s + 17 * 60 + 47 + 33; //表示是否迟到
7          int need2 = h * 3600 + m * 60 + s + 24 * 60 + 47 + 12 + 33; //表示能否
            吃到早餐
8          if( need1 < tmp ) printf("haoye\n");
9          else printf("haofan\n");
10         if( need2 < tmp ) printf("haoye haoye\n");
11         else printf("haoe~\n");
12     }
13     return 0;
14 }

```

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：

```
1  #include<stdio.h>
2  int main(){
3      long long n;
4      while(~scanf("%lld",&n)){
5          long long t1,t2,Max = -1,mark = 10;
6          while (n >= mark) {
7              t1 = n % mark;
8              t2 = n / mark;
9              mark *= 10;
10             if ( (t1 + t2) > Max ) Max = t1 + t2;
11         }
12         printf("%lld\n", Max);
13     }
14     return 0;
15 }
```

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：

```
1  #include<stdio.h>
2  #include<string.h>
3  long long a[100010],b[100010];
4  int main(){
5      int T;
6      scanf("%d",&T);
7      a[1] = b[1] = 1;
8      for (int i = 2; i <= 50; ++i) {
9          a[i] = b[i - 1];
10         b[i] = a[i - 1] + b[i - 1];
11     }
12     while (T--) {
13         int n;
14         scanf("%d",&n);
15         printf("%lld %lld\n",a[n],b[n]);
16     }
17     return 0;
18 }
```

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。

```
1  #include<stdio.h>
2  int n,k;
3  bool check(int n){
4      int cou=0;
5      while(n){
6          if(n%10==3)++cou;
7          n/=10;
8      }//逐位计算有几个3
9      if(cou==k)return 1;
10     else return 0;
11 }
12 int main(){
13     while(~scanf("%d %d",&n,&k)){
14         for(;;n++){
15             if(check(n)){
16                 printf("%d\n",n);
17                 break;
18             }
19         }
20     }
21 }
```

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。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int mod=1e9+7;
5  const int N=1e5+7;
6
7  int n,l,x,m;
8  char c[N];
9
10 int main() {
11     while(~scanf("%d",&l)) {
12         getchar();
13         for(int i=1; i<=l; i++) scanf("%c",&c[i]);
14         scanf("%d",&n);
15         while(n--) {
16             scanf("%d",&x);
17             for(int i=x,j=l-x+1;i<=j; i++,j--) {
18                 char t = c[i];
19                 c[i] = c[j];
20                 c[j] = t;
21             }
22         }
23         for(int i=1;i<=l;i++) printf("%c",c[i]);
24         puts("");
25     }
26 }
```

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

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int mod=1e9+7;
5  const int N=1e5+7;
6  int n,l,x,nn;
7  int c[N];
8  int f[N];
9
10 int main() {
11     while(~scanf("%d",&l)) {
12         memset(f,0,sizeof(f));    //把f数组全部置为0
13         memset(c,0,sizeof(c));    //把c数组全部置为0
14         getchar();
15         for(int i=1; i<=l; i++) scanf("%c",&c[i]);
16         scanf("%d",&n);
17         for(int i=0; i<n; i++) {
18             scanf("%d",&x);    //统计每个位置出现了多少次
19             f[x]++;
20         }
21         for(int i=1; i<=l/2; i++) {
22             f[i]+=f[i-1];
23         }
24         for(int i=1; i<=l/2; i++) {
25             if(f[i]%2!=0) swap(c[i],c[l-i+1]);
26         }
27         for(int i=1; i<=l; i++) printf("%c",c[i]);
28         puts("");
29     }
30 }
```

