进制转换

本节我们来学习关于进制转换的问题，主要包括十进制转成任意进制和任意进制转成十进制。本节内容相对简单，通过本节的学习，还有一个目的，就是巩固大家对函数的应用。

位权

我们说一个数值，比如 321，这是它的值，值是不变的，但是表示 321 这个值的数可以是多种形式，比如十进制形式，比如二进制形式（101000001）$_2$，比如十六进制形式（141）$_{16}$。在这里，我们要理解数和值分开的概念，值是不变的，数可以是多种形式，一个值可以用多种数来表示。当然，为了描述这个值，我们一般采用十进制的形式表示。对于不同的数（进制形式不同），每一位都有一个位权。

十进制：321=3*100+2*10+1*1。

二进制：321=（101000001）$_2$=1*256+0*128+1*64+0*32+0*16+0*8+0*4+0*2+1*1

十六进制：321=（141）$_{16}$=1*32+4*16+1*1

每一个进制数中的每一位，都有一个位权，对于 k 进制数的第 i 位，它的位权为 k^(i-1)，如对于 10 进制的第 3 位的位权为 10^(3-1)=10^2=100，即十进制数的第三位的位权为 100。

十进制转任意进制

还记得数位分离吗？对于给定的十进制数 x，我们分离它的个位是 X%10，去除它的个位是 x/10,数位分离的代码是这样的：

```
#include<iostream>
using namespace std;
int main(){
    int x;
    cin>>x;
    while(x!=0){
        cout<<x%10<<" ";
        x/=10;
    }
}
```

其实对于这个程序，正确的理解方式是对于一个值 x，将其转换成十进制输出。当然，数位分离出来的数值,是倒序的,我们可以用一个数组存储分离出的每一位,然后倒序输出即可，如下：

```
#include<iostream>
using namespace std;
int a[100],len,x;
int main(){
    cin>>x;
    while(x!=0){
        a[++len]=x%10;//len 表示数组的长度
        x/=10;
    }
    for(int i=len;i>=1;i--){//倒序输出
        cout<<a[i];
    }
}
```

对于二进制数 x，将 x%2，可以得到它的低位，x/2 可以去掉低位，分离出二进制数 x 的每一位，也可以类似十进制的分离方式，只是底数要换成 2。以下代码实现十进制转二进制数：

```
#include<iostream>
using namespace std;
int a[100],len,x;
int main(){
    cin>>x;
    while(x!=0){
        a[++len]=x%2;//基数为 2
        x/=2;//基数为 2
    }
    for(int i=len;i>=1;i--){//倒序输出
        cout<<a[i];
    }
}
```

这个代码跟上个代码基本没变化，这种转换的方法，我们可以概括为**除 2 取余倒取法**。当然，不独转换成二进制数是这样，转换成 k 进制数，可以用**除 k 取余倒取法**。

例 1：输入一个十进制数 n 和一个整数 k(2<=k<=16)，将 n 转换成 k 进制数然后输出。
分析：
可以用除 k 取余倒取法。在输出的时候，如果是大于 10 的余数，需要表示为大写字母。
代码：

```
#include<iostream>
using namespace std;
int a[100],len,n,k;
int main(){
    cin>>n>>k;
    while(n!=0){
```

```
        a[++len]=n%k;
        n/=k;
    }
    for(int i=len;i>=1;i--){
        if(a[i]<10)cout<<a[i];
        else cout<<(char)(a[i]-10+'A');
    }
}
```

任意进制转十进制

首先，我们来看二进制转十进制的情况，这种情况等价于求二进制数对应的值。我们来看一个例子：

$(11011)_2=1*16+1*8+0*4+1*2+1*1=27$

在这里，转换的方法是求出二进制数每一位的位权，然后将位权乘以位数，再逐项相加，即可得到对应的数。

```
#include<iostream>
#include<string>
using namespace std;
string s;
int n,pow=1;
int main(){
    cin>>s;//考虑二进制数会很长，超过 int 存储范围，在此用字符串读入
    for(int i=s.size()-1;i>=0;i--){//倒序，从低位开始计算
        n+=(s[i]-'0')*pow; //位数乘以位权，然后累加
        pow*=2;//计算下一位的位权
    }
    cout<<n;
}
```

例 2：输入一个进制数和一个整数 k，k 表示输入的数属于 k 进制数，将该 k 进制数转换成十进制数输出。

分析：

对于任意进制，也可以用类似的加权法转换成十进制数，但是权数的基数为 k。在读入时，考虑其他进制可能含有大写字母或者低于十进制的长度较长，在此使用字符串读入，使用字符串也方便逐位枚举。

代码：

```
#include<iostream>
```

```
#include<string>
using namespace std;
string s;
int n,pow=1,k;//k 表示读进的数是 k 进制数
int main(){
    cin>>s>>k;//超过十进制数会带有字母，所以用字符串读入
    for(int i=s.size()-1;i>=0;i--){//倒序，从低位开始计算
        int x;//求出该字符对应的数值
        if(s[i]>='0'&&s[i]<='9')x=s[i]-'0';
        else x=s[i]-'A'+10;
        n+=x*pow; //位数乘以位权，然后累加
        pow*=k;//累乘 k，计算下一位的位权
    }
    cout<<n;
}
```

在一些文献中，任意进制转十进制数，还有另一种写法，不需倒序处理，每次移位处理即可，代码如下，注意揣摩异同：

```
#include<iostream>
#include<string>
using namespace std;
string s;
int n,k;
int main(){
    cin>>s>>k;
    for(int i=0;i<s.size();i++){  //顺序处理
        int x;
        if(s[i]>='0'&&s[i]<='9')x=s[i]-'0';
        else x=s[i]-'A'+10;
        n=n*k+x;//关键理解这一句，n*k 相当于左移一位再加上 x
    }
    cout<<n;
}
```

以上代码的优势在于对于输入的数据，可以在线处理。

==========网站练习==========

斜二进制数

【题目描述】
当一个数字是十进制数，每项的基数表现为 10 的 K 次方。（数字是有限的，从右边到左边，

在最末的数字是 10 的 0 次方)。 举例来说,
81307(10) = 8 * 10^4 + 1 * 10 ^3 + 3 * 10^2 + 0 * 10^1 + 7 * 10^0
= 80000 + 1000 + 300 + 0 + 7
= 81307.
当一个数字是二进制数, 每项的基数表现为 2 的 k 次方。 举例来说,
10011(2) = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0
= 16 + 0 + 0 + 2 + 1
= 19.
在斜二进制中, 每项的基数表现 2 的(k+1)次方减 1。举例来说,
10120(skew) = 1 * (2^5-1) + 0 * (2^4-1) + 1 * (2^3-1) + 2 * (2^2-1) + 0 * (2^1-1)
= 31 + 0 + 7 + 6 + 0
= 44.
例如: 最初 10 个数字在斜的二进制中是 0,1,2,10,11,12,20,100,101, 和 102 。
【输入】
输入文件包括一行数据, 一个斜二进制整数。
【输出】
输出斜二进制数字的十进制值, 要是超过 2147483647, 则输出“too long!”
【输入样例 1】
11
【输入样例 2】
10120
【输出样例 1】
4
【输出样例 2】
44
【分析】
本题的模型是将二进制转成十进制, 但有点不同的是, 斜二进制的位权和二进制不同, 利用加权法可以轻松实现本题。
【代码】

```
#include<iostream>
#include<string>
using namespace std;
string s;
long long n,pow=1;
int main(){
    cin>>s;
    for(int i=s.size()-1;i>=0;i--){
        pow*=2;
        n+=(s[i]-'0')*(pow-1);  //pow-1 为该位的位权
        if(n>2147483647){
            cout<<"too long!";
            return 0;
        }
```

```
    }
    cout<<n;
}
```

特别的十进制数

【题目描述】
特别的十进制数是指具有如下特征的数：它的各位数字之和等于该数的 16 进制表示的各位数字之和，并且还等于该数的 12 进制表示的各位数字之和。
例如，2991 的各位数字之和为 2＋9＋9＋1＝21，它的 12 进制表示是 189312，各位数字之和也是 21。但是 2991 的 16 进制表示是 BAF16，并且 11＋10＋15＝36，所以 2991 不具备指定的特征。
又如，2992 在全部三种表示法中（包括 BB016）各位数字之和都是 22，所以 2992 具备指定的特征。故 2992 是一个特别的十进制数。
【输入】
输入文件只有一个长度不超过 9 位的十进制正整数。
【输出】
第一行为输入的十进制数所对应的十六进制数各位数字之和，第二行为分"Yes"（具备特征）或"No"（不具备特征）。
【输入样例 1】
3
【输入样例 2】
112
【输出样例 1】
3
Yes
【输出样例 2】
7
No
【分析】
根据题意，将十进制数转成十二进制数、十六进制数，同时求数制下的位值和即可。
【代码】

```
#include<iostream>
using namespace std;
int n;
int getsum(int n,int k){  //n 转成 k 进制，求 k 进制下的位数和
    int sum=0;
    while(n!=0){
        sum+=n%k;
        n/=k;
    }
    return sum;
}
```

```
int main(){
    cin>>n;
    int a=getsum(n,10);
    int b=getsum(n,12);
    int c=getsum(n,16);
    cout<<c<<endl;
    if(a==b&&a==c){
        cout<<"Yes"<<endl;
    }else{
        cout<<"No"<<endl;
    }
}
```

Niven 数

【题目描述】
一个 Niven 数就是一个它的各位数字之和能整除它本身的数，
例如:111 是 Niven 数。因为 111 mod (1+1+1)=0。
同样地， 在其他数制，我们仍能有这样的 Niven 数。给出 数制 b (2<=b<=10) 和一个 b 数制的数,你要判断这个数是否是 Niven 数。输入数据包含多组数据。
【输入】
每个块包含若干测试数据。
每个测试数据占一行，先是数制 b，之后是一个合法的数制中的数。
b 为 0 表示测试块结束。
【输出】
如果该数在该数制为 Niven 数，就输出"yes"，否则输出"no"。
两个测试块之间用空行隔开。
【样例输入】
10 111
2 110
10 123
6 1000
8 2314
0
【样例输出】
yes
yes
no
yes
no
【分析】
求将 b 数制转成十进制，同时求 b 数制的位置和，判断能否整除。

【代码】

```
#include<iostream>
#include<string>
using namespace std;
int b;
string s;
int getvalue(string s,int k){  //将 k 进制 s 转成十进制返回
    int n=0;
    for(int i=0;i<s.size();i++){
        n=n*k+s[i]-'0';
    }
    return n;
}
int getsum(string s){ //获得 s 的数位和返回
    int sum=0;
    for(int i=0;i<s.size();i++){
        sum+=s[i]-'0';
    }
    return sum;
}
int main(){
    while(cin>>b){
        if(b==0)break;
        cin>>s;
        if(getvalue(s,b)%getsum(s)==0){
            cout<<"yes"<<endl;
        }else{
            cout<<"no"<<endl;
        }
    }
}
```

回文数

【题目描述】

我们说一个数字如果从左到右和从右到左是一样得就是回文数。例如 75457 就是一个回文数。当然，这性质还依赖它的进制。17 在十进制下不是回文数，不过在二进制下就是回文数了(10001) 。现在就是要一个数在 2 到 16 进制下是否回文数。

【输入】

输入数据由多个整数构成。每个给出的数字 0 < n < 50000 各占一行，并且都是以十进制形式给出。输入以零为结束。

【输出】

你的程序要输出令所给数 I 是回文数的进制。如果这个数在 2 到 16 进制里都不是回文数，

你的程序就要输出这个数不是回文数的信息。

【样例输入】

17

19

0

【样例输出】

Number 17 is palindrom in basis 2 4 16

Number 19 is not a palindrom

【分析】

枚举一个数的各种数制（2 到 16 进制），分别判断在该数制下是否为回文数，利用数组标记起来，输出时结合数组标记进行输出。

【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
int n,cnt;
bool f[17];
bool check(int n,int k){
    //将 n 转成 k 进制
    int a[20],len=0;
    while(n!=0){
        a[++len]=n%k;
        n/=k;
    }
    //判断 n 的 k 进制形式是不是回文数，
    //判断的方式是首尾逐位比较，循环到长度的一半
    for(int i=1;i<=len/2;i++){
        if(a[i]!=a[len-i+1])return false;
    }
    return true;
}
int main(){
    while(cin>>n){
        if(n==0)break;
        memset(f,false,sizeof(f));
        cnt=0;
        //在 2 至 16 进制下检验 n 是否为回文数
        for(int i=2;i<=16;i++){
            if(check(n,i)){ //是回文数则数组标记
                f[i]=true;
                cnt++;  //cnt 用来标记回文数的个数
            }
        }
        cout<<"Number "<<n<<" is ";
```

```
        if(cnt==0){
            cout<<"not a palindrom"<<endl;
        }else{
            cout<<"palindrom in basis";
            for(int i=2;i<=16;i++){
                if(f[i])cout<<" "<<i;
            }
            cout<<endl;
        }
    }
}
```

致命的珠宝

【题目描述】
Mini 使用神风无影来到了大魔王所居住的洞穴（FAQ：干嘛不直接飞进去？ANS：那样太无聊了，故事没情节 TUT…），门口却有着险恶的机关。门上有着 N 个宝珠，每个宝珠都有一个数字。传说，只要宝珠里的两颗珠撞在一起后就会发出奇异的光彩，但发出的光彩有可能是致命的，也有可能是打开前进之路的钥匙。Mini 询问老者后，得知要想打开这扇门，就得找出两颗珠宝，使这两颗珠宝撞在一起后产生的能量值最接近 123。
两颗珠宝撞在一起以后产生的能量值的计算方法是：将两个珠宝所代表的数字转换为 7 进制的数后，一一对照这两个七进制数的每一位，若相同，则结果为 0 否则为 1。
如：两颗珠子所代表的数为 18 和 370，将这两个数转化为 7 进制后是 24 和 1036，对于高位不足的数，采取高位添‘0’的方法，即两个数为 0024，1036。最后得到的能量值 C 为 1011，再将 C 当作二进制数转换为十进制数。那么转换后的 C 就为这两个珠撞在一起以后所产生的能量值。
【输入】
第一行一个数 N，表示宝珠的数量。（2<=N<=900）
第二行 N 个数，每个数用空格隔开，每个数表示第 I 个宝珠所代表的数字(0<=每个数<=11111)
【输出】
一个数，代表你所找到的最接近 123 的能量值
【样例输入】
5
18 370 45 36 78
【样例输出】
15
【提示】
370 和 78 这两颗宝珠所产生的能量值 15 最接近 123
第一题中：如果两个宝珠的能量值为 122，另两个宝珠的能量值为 124，则用 I 比较小的那个。如果 I 相同则用 J 比较小的那个 I、J 表示两颗珠子是第几个珠。均为非负整数
【分析】
枚举任意两颗珠子，将这两个珠子按照题意转成规则求出能量值 v，将 v 与当前最优值 best 比较谁最靠近 123，更新 best 的值。
【代码】

```cpp
#include<iostream>
#include<cstring>
#include<cmath>
using namespace std;
int a[901],n,best;

void turn7(int x,int a[],int & len){  //将 x 转成 7 进制，存在数组 a，注意参数用法
    len=0;
    while(x!=0){
        a[++len]=x%7;
        x/=7;
    }
}

int getvalue(int x,int y){
    int a[7],b[7],c[7],len1,len2,len;
    memset(a,0,sizeof(a));  //数组初始化
    memset(b,0,sizeof(b));
    memset(c,0,sizeof(c));
    turn7(x,a,len1);  //将 x 转成 7 进制，存在 a 数组
    turn7(y,b,len2);  //将 y 转成 7 进制，存在 b 数组
    len=max(len1,len2);
    for(int i=1;i<=len;i++){  //将 a 和 b 逐位比较，赋值给 c 数组
        if(a[i]!=b[i])c[i]=1;
    }
    int n=0;
    for(int i=len;i>=1;i--){  //将 c 数组存储的二进制数转成十进制数
        n=n*2+c[i];
    }
    return n;
}

int main(){
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i];
    for(int i=1;i<n;i++){  //两重循环枚举任意两个数
        for(int j=i+1;j<=n;j++){
            int v=getvalue(a[i],a[j]);  //求两个数产生的能量值，赋给 v
            if(abs(v-123)<abs(best-123)){  //v 与当前最优值 best 比较
                best=v;
            }
            if(best==v&&v<best){  //相等情况取小的
                best=v;
            }
```

```
45.         }
46.     }
47.     cout<<best;
48. }
```

猫猫的小鱼

【题目描述】
猫猫是丛林里很多动物心中的天使，她为此十分自豪。猫猫最爱吃鱼了，她每天都要去池塘钓鱼吃。猫猫经常吃鱼脑，数学特别强，然而，小女生的性格决定了她的贪玩。
一天，猫猫钓到了很多条鱼。她并不想马上就把可怜的鱼儿吃掉，而是先折磨够之后再吃（有句话叫什么来着～最毒不过猫猫心）。
猫猫将这很多很多（数不过来）条鱼按照外观的漂亮程度排序，每个鱼的编号依次为 1、2、3……N，第 i 条鱼的美观程度为 3^(i-1)。
猫猫要把这些鱼放到桶里去。她每次拿的鱼的数目是任意的。鱼的“总美观程度”为各条鱼美观程度之和。例如：猫猫这一次拿了第一条鱼和第三条鱼，那么美观程度为 1+9=10。
猫猫想知道，她可以获得的第 k 大的“总美观程度”是多少。
读入 k，输出猫猫能够获得的，第 k 大的“总美观程度”。
【输入】
数据包含 n+1 行，第一行读入 n(n<=100)。以下 n 行每行包含一个 k。
【输出】
输出包含 n 行，每行输出一个对应的结果。
【样例输入】
1
7
【样例输出】
13
【提示】
猫猫能够拿到的美观程度从小到大为 1、3、4、9、10、12、13……所以第 7 大的美观程度是 13。
对于 50%的输入文件，有 k≤5000。
对于 100%的输入文件，有 k≤2^31-1。
【分析】
我们注意到题目中所求的递增数列中的每个数都是整数，且每个数字都是由不同的 3 的整数次幂的和构成的，所以这个数字的三进制表示中只有 0 或 1，我们注意到二进制也是由 0 和 1 组成的，结合题目中要我们求第 k 小的数字，因此不难想到在全体整数与数列之间建立一个一一对应，使得数字 k 的二进制表示与它所对应的三进制表示相同。举例说明：

K 的值	对应的二进制	对应关系	相应的三进制	第 k 大的美观程度
1	1		1	1
2	10		10	3
3	11		11	4
4	100		100	9

5	101		101	10
6	110		110	12
7	111		111	13

因此我们需要做的就是把 k 这个数字转换成二进制，然后再把相应的二进制当作三进制转换成十进制就可以了。 注意数据可能较大，用 long long。

【代码】

```
#include<iostream>
#include<cstring>
using namespace std;
int n,a[50],len;  //因为 k 多达 10 位数，转成二进制可能达到 40 位，所以数组 a 开了 50

void turn2(long long x){  //将 x 转成 2 进制
    memset(a,0,sizeof(a));
    len=0;
    while(x!=0){
        a[++len]=x%2;
        x/=2;
    }
}
long long turn3(){  //求二进制对应的三进制的值
    long long sum=0,pow=1;
    for(int i=1;i<=len;i++){
        sum+=a[i]*pow;
        pow*=3;
    }
    return sum;
}

int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        long long x;
        cin>>x;
        turn2(x);  //将 x 转成二进制
        cout<<turn3()<<endl;  //输出二进制对应的三进制的值
    }
}
```