进制转换

一、进制转换方法：

1、将P进制转换为十进制的方法:

设一个 P 进制数 x 为 $a_1a_2a_3...a_n$，则可以转换为十进制数
$y = a_1 * P^{n-1} + a_2 * P^{n-2} + ... + a_{n-1} * P + a_n$，这个公式可以通过**循环实现**：

```
int y = 0, product = 1;
while(x != 0){
    y = y + (x % 10) * product;      // x%10 用于获取x的个位数
    x = x / 10;                      // x/10 用于去掉x的个位数
    product = product * P;
}
```

2、将十进制转换为Q进制的方法：除基取余法

每次将待转换数 x 除以 Q， 然后将得到的**余数**作为低位存储，**商**则继续除以Q并重复之前的操作。这个方法可通过循环实现：

```
int z[40], num = 0;
do{
    z[num++] = y % Q;        //除基取余
    y = y / Q;
}while(y != 0);              //当商不为0 时进行循环
```

二、甲类题目

1、General Palindromic Number

A number that will be the same when it is written forwards or backwards is known as a **Palindromic Number**.

For example, 1234321 is a palindromic number. **All single digit numbers are palindromic numbers.**

Although palindromic numbers are most often considered in the **decimal system**, the concept of **palindromicity** can be applied to the natural numbers in **any numeral system.**

Consider a number $N > 0$ in base $b \geq 2$, where it is written in standard **notation** with $k + 1$ digits a_i as $\sum_{i=0}^{k}(a_i * b^i)$.

Here, as usual, $0 \le a_i < b$ for all i and a_k is non-zero. **Then N is palindromic if and only if $a_i = a_{k-i}$ for all i.**

Zero is written 0 in any base and **is also palindromic** by definition.

Given any positive decimal integer *N* and a base *b*, you are supposed to tell if *N* is a palindromic number in base *b*.

Input Specification:

Each input file contains one test case. Each case consists of two positive numbers *N* and *b*, where $0 < N \le 10^9$ is the decimal number and $2 \le b \le 10^9$ is the base. The numbers are separated by a space.

Output Specification:

For each test case, first print in one line `Yes` if *N* is a palindromic number in base *b*, or `No` if not.

Then in the next line, print *N* as the number in base *b* in the form "$a_k a_{k-1} \ldots a_0$". Notice that there must be no extra space at the end of output.

Sample Input:

```
27 2
```

Sample Output:

```
Yes
1 1 0 1 1
```

Sample Input 2:

```
121 5
```

Sample Output 2:

```
No
4 4 1
```

Key Codes:

```
bool Judge(int z[], int num){
    if(num == 1) return true;           //任何单个数字都是回文数
    for(int i=0;i != num-1-i;i++){
        if(z[i] != z[num-1-i]) return false;
    }
    return true;
```

```
}
int main(){
    int N, b,num=0, z[40] = {0};
    scanf("%d %d", &N, &b);
    do{
        z[num++] = N % b;
        N = N / b;
    }while(N != 0);
    bool flag = Judge(z, num);

    if(flag) printf("Yes\n");
    else printf("No\n");

    for(int i=num-1;i>=0;i--){
        if(i != 0) printf("%d ", z[i]);
        else printf("%d", z[i]);
    }
    return 0;
}
```

2、Colors in Mars

People in Mars represent the colors in their computers in a similar way as the Earth people. That is, a color is represented by a 6-digit number, where **the first 2 digits are for** `Red`**, the middle 2 digits for** `Green`**, and the last 2 digits for** `Blue`.

The only difference is that they use **radix 13 (0-9 and A-C) instead of 16.** Now given a color in three decimal numbers (each between 0 and 168), you are supposed to output their Mars RGB values.

Input Specification:

Each input file contains one test case which occupies a line containing the three decimal color values.

Output Specification:

For each test case you should output the Mars RGB value in the following format: **first** output `#`, **then** followed by a 6-digit number where all the English characters must be upper-cased. If a single color is only 1-digit long, you must print a `0` to its left.

Sample Input:

```
15 43 71
```

Sample Output:

```
#123456
```

Key Codes:

```
int Exchange(int X, int x[]){
    int num=0;
    do{
        x[num++] = X % 13;
       X = X / 13;
    }while(X != 0);
    return num;
}
int main(){
    int R, G, B;
    int rgb[3][3], num[3];
    scanf("%d %d %d", &R, &G, &B);
    num[0] = Exchange(R, rgb[0]);
    num[1] = Exchange(G, rgb[1]);
    num[2] = Exchange(B, rgb[2]);
    printf("#");
    for(int i=0;i<3;i++){
        if(num[i] == 1){
            if(rgb[i][0] == 10){
                printf("0A");
            }
            else if(rgb[i][0] == 11){
                printf("0B");
            }
            else if(rgb[i][0] == 12){
                printf("0C");
            }
            else{
                printf("0%d", rgb[i][0]);
            }
            continue;
        }
        for(int j=num[i]-1;j>=0;j--){
            if(rgb[i][j] == 10){
                printf("A");
            }
            else if(rgb[i][j] == 11){
                printf("B");
            }
            else if(rgb[i][j] == 12){
                printf("C");
            }
            else{
                printf("%d", rgb[i][j]);
```

```
            }
        }
    }
    return 0;
}
```

代码优化：

```
char radix[13] = {'0', '1', '2','3', '4','5','6', '7', '8', '9', 'A', 'B',
'C'};
int main(){
    int r,g,b;
    scanf("%d%d%d", &r, &g, &b);
    pritnf("#");
    printf("%c%c", radix[r/13], radix[r%13]);
    printf("%c%c", radix[g/13], radix[g%13]);
    printf("%c%c", radix[b/13], radix[b%13]);
    return 0;
}
```

3、A+B in Hogwarts

If you are a fan of Harry Potter, you would know the world of magic has its own **currency system** -- as Hagrid explained it to Harry, "**Seventeen** silver Sickles to a Galleon and **twenty-nine** Knuts to a Sickle, it's easy enough."

Your job is to write a program to compute *A*+*B* where *A* and *B* are given in the standard form of `Galleon.Sickle.Knut` (`Galleon` is an integer in $[0, 10^7]$, `Sickle` is an integer in [0, 17), and `Knut` is an integer in [0, 29)).

Input Specification:

Each input file contains one test case which occupies a line with *A* and *B* in the standard form, separated by one space.

Output Specification:

For each test case you should output the sum of *A* and *B* in one line, with the same format as the input.

Sample Input:

```
3.2.1 10.16.27
```

Sample Output:

```
14.1.28
```

一种错误的解决方案：

P[0] 、 A[0] 最大为10^7， 因此， $(P[0]+A[0])*493$ 会超出 int 类型数据的表示范围

```
int main(){
    int P[3], A[3], sum[3] = {0};
    scanf("%d.%d.%d %d.%d.%d", &P[0], &P[1], &P[2], &A[0], &A[1], &A[2]);
    int sum_K = (P[0]+A[0])*493 +(P[1]+A[1])*29 + P[2] + A[2];
    sum[0] = sum_K / 493;
    sum[1] = (sum_K - sum[0]*493) / 29;
    sum[2] = sum_K % 29;
    printf("%d.%d.%d", sum[0], sum[1], sum[2]);
    return 0;
}
```

Key Codes：

```
int main(){
    int P[3], A[3], sum[3] = {0};
    scanf("%d.%d.%d %d.%d.%d", &P[0], &P[1], &P[2], &A[0], &A[1], &A[2]);

    sum[2] = (P[2]+A[2]) % 29;      //2号位的结果
    int carry = (P[2]+A[2]) / 29;   //进位
    sum[1] = (P[1]+A[1]+carry) % 17;//1号位的结果
    carry = (P[1]+A[1]+carry) / 17;
    sum[0] = P[0]+A[0]+carry;       //0号位的结果

    printf("%d.%d.%d", sum[0], sum[1], sum[2]);
    return 0;
}
```