Chem Is Another Try!——罗仪 & Arthur

题目描述

炼金术师 Lucifer•Tang 又要进行新的实验了。实验很危险，她要格外小心。因此为了采取一些安全措施，Tang 大魔决定计算前四周期元素的元素的能级排布式，与拖延症艰苦作战的她便将这一任务交给了你。

输入

多组测试数据，每组测试数据为一个整数 n(1≤n≤36)，为元素的原子序数。

输出

对于每组数据，输出两行。
第一行为该元素的化学符号
第二行为该元素的能级排布式。
具体见样例

输入样例

1
17
24

输出样例

H
1s1
Cl
1s22s22p63s23p5
Cr
1s22s22p63s23p63d54s1

HINT

能级：s、p、d（本题只需要用到这些…）s 能级最多填充两个电子，p 能级最多填充6个电子，d 能级最多填充10个电子，填充时按照能级能量从低到高填充,相同能级内电子能量相同。（也就是说把低能级填满才可以填充更高能级）
能级能量从低到高排列为：1s<2s<2p<3s<3p<4s<3d<4p
撰写规则:从左到右，按照能级能量，从低到高依次填写,比如对于氯(Cl)，1s 上填充为2个，

2s 上填充为2个，2p 上填充为6个，3s 上填充为2个，3p 上填充为5个。因此能级排布式为：1s22s22p63s23p5（意思就是1s2-2s2-sp6-3s2-3p5）
此外，第四周期有两个例外，究竟是什么呢？
这个可以度娘呦0w0

Analysis：

既然要输出元素符号，我开始想设置一个字符数组，把元素符号放进去。后来发现输出时会只能输出一个字节。比如应该输出“Cl”，但实际上会输出”l”.这需要大家注意一下。因为字符数组中的一个元素存放一个字符，它在内存中占用一个字节。所以只能用字符串了。普及单双引号的区别："a"和'a'的区别，前者是字符串，后者是字符。因此 case(1): return "H";中双引号若写成了单引号系统会报错的。

用 switch 会比 if 稍微快些吧~

输出能级时需要百度一下，看到百科里的前 36 号元素，对比一下区别与规律，就能写出代码了。注意例外的几个元素。

Code for Reference：

```
#include <iostream>
#include<cstring>
using namespace std;
string f(int);
int main()
{
	int n;
	while(cin>>n)
	{   cout<<f(n)<<endl;
	    if(n<=2) cout<<"1s"<<n<<endl;
	    else if(n>2&&n<=4)
	        cout<<"1s22s"<<n-2<<endl;
	    else if(n>4&&n<=10) cout<<"1s22s22p"<<n-4<<endl;
	    else if(n>10&&n<=12) cout<<"1s22s22p63s"<<n-10<<endl;
	    else if(n>12&&n<=18) cout<<"1s22s22p63s23p"<<n-12<<endl;
	    else if(n>18&&n<=20)
```

```
                cout<<"1s22s22p63s23p64s"<<n-18<<endl;
            elseif(n>20&&n<30&&n!=24&&n!=29)
cout<<"1s22s22p63s23p63d"<<n-20<<"4s2"<<endl;
            else if(n==24) cout<<"1s22s22p63s23p63d54s1"<<endl;
            else if(n==29) cout<<"1s22s22p63s23p63d104s1"<<endl;
            else if(n==30) cout<<"1s22s22p63s23p63d104s2"<<endl;
            else    cout<<"1s22s22p63s23p63d104s24p"<<n-30<<endl;
      }
}
string f(int l)
{
      switch(l)
            {
            case(1): return "H";
            case(2): return "He";
            case(3): return"Li";
            case(4): return "Be";
            case(5): return"B";
            case(6): return"C";
            case(7): return"N";
            case (8): return"O";
            case (9): return"F";
            case (10): return"Ne";
            case (11): return"Na";
            case (12): return"Mg";
            case (13): return"Al";
            case (14): return"Si";
            case (15): return"P";
            case (16): return"S";
            case (17): return"Cl";
            case (18): return"Ar";
            case (19): return"K";
            case (20): return"Ca";
            case (21): return"Sc";
            case (22): return"Ti";
            case (23): return"V";
            case (24): return"Cr";
            case (25): return"Mn";
            case (26): return"Fe";
            case (27): return"Co";
            case (28): return"Ni";
            case (29):return"Cu";
            case (30): return"Zn";
            case (31): return"Ga";
```

```
            case (32): return"Ge";
            case (33): return"As";
            case (34): return"Se";
            case (35): return"Br";
            case (36): return"Kr";
            }
}
```

官方分析：

/*55555555555555 你们这些坏淫，居然都在打表 QAQ*/
言归正传，其实打表也不失为一种策略，在出题的时候也考虑到了这一点。不过还是希望大家对本题进行一下深层次的思考。

规则其实也很简单，每个能级能够填充的上限已经是确定的了，这个可以存在一个数组里，而每个能级的名称也是确定的，这个也可以存储在一个数组里，每个元素的名称是确定的，这个也可以存储在一个数组里。
最后，使用一个数组，来记录对应元素在每一个能级上的电子填充个数，如果当前剩余电子个数不小于要填充的能级所能容纳的上限，则将这一能级填满，电子个数减去该上限，并移向下一能级继续填充，如果出现 Cr 与 Cu，则只需要在计算完成之后，将 4s 上的电子减去一个，将 3d 上的电子增加一个即可。

参考代码

```
#include<iostream>
using namespace std;
const string c[8]={"1s","2s","2p","3s","3p","3d","4s","4p"};   //能级名称
string
element[37]={"WHAT?","H","He","Li","Be","B","C","N","O","F","Ne","Na","Mg","Al","Si","P","S","
Cl","Ar","K","Ca","Sc","Ti","V","Cr","Mn","Fe","Co","Ni","Cu","Zn","Ga","Ge","As","Se","Br","Kr"};
//元素名称，没错，字符串也可以开数组
int num[8],n,maxNum[8]={2,2,6,2,6,10,2,6};
//num 数组存储元素在各个能级里填充的电子个数
//maxNum 存储每个能级最多容纳的电子个数
void Init()     //初始化 num 数组，将所有元素置为 0
{
       for(int i=0;i<8;i++)
           num[i]=0;
}
void comPare(int &n,int i)   //比较当前剩余电子个数与当前能级的电子上限
{      //请注意这里是按引用传递，也就是说下文中的 n0 在每次调用该函数后，结果会改变
      if(n>=maxNum[i])     //如果剩余的不小于上限
      {
           num[i]=maxNum[i];   //填满该能级
           n-=maxNum[i];          //剩余电子个数应减去这一上限
      }
      else if(n>0)
      {
           num[i]=n;     //否则直接将所有剩余电子填入该能级
           n=0;
      }
}
void display(int n)     //输出最终结果
{
        cout<<element[n]<<endl;
        for(int i=0;i<8;i++)     //由于设置的时候 4s 是在 3d 的前面，因此一定要全部遍历
            if(num[i]!=0)          //如果当前能级不为空，则输出
               cout<<c[i]<<num[i];
       cout<<endl;
}
void Manipulate(int n)     //计算最终结果的函数
{
       int i=0,n0=n;     //使用一个临时变量记录 n
       while(i<5&&n0>0)       //前五个能级没有特殊情况，统一填充
       {
```

```
            comPare(n0,i);   //对这五个能级每个能级进行填充
            i++;             //再转向下一能级
        }
        comPare(n0,6);comPare(n0,5);comPare(n0,7);   //接下来的顺序应该是 4s、3d、4p
        if(n==24||n==29)      //计算完成后，这里特别判断一下 Cr 与 Cu
        {
            num[5]++;    //如果是这两个则 4s 上的-1，3d 上的+1
            num[6]--;
        }
}
int main()
{
    while(cin>>n)
    {
        Init();
        Manipulate(n);
        display(n);
    }
}
```