1234：2011

题目描述

已知长度最大为200位的正整数n，请求出2011^n的后四位。

输入

第一行为一个正整数k，代表有k组数据（k≤200），接下来的k行，每行都有一个正整数n，n的位数≤200。

输出

每一个n的结果为一个整数占一行，若不足4位，去除高位多余的0。

输入样例

```
3
5
28
792
```

输出样例

```
1051
81
5521
```

解析

这道题一上来会给人一种错觉，可以使用快速幂。但是，稍加分析我们就能发现n的范围是长度为100的数字，很明显，在常规的运算中任何数据类型都无法满足这样巨大的计算？难道是高精度的快速幂？

淡定，来看看下面的分析图，我们分别计算了2和3的不同幂次方，并对5取余，结果如下。

很明显，我们计算出来的数据是呈循环状态分布的。

2的幂数	0	1	2	3	4	5	6	7	8	9	10	11	12	13
结果	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192
对5取余	1	2	4	3	1	2	4	3	1	2	4	3	1	2

3的幂数	0	1	2	3	4	5	6	7	8	9	10	11
结果	1	3	9	27	81	243	729	2187	6561	19683	59049	177147
对5取余	1	3	4	2	1	3	4	2	1	3	4	2

上图的示例是对5取余得出的结果，那么本题是求后四位，即对10000取余，那这样的计算结果是否也依然符合上面的规律呢？答案是肯定的。因此，对于本题，我们只需要找到循环的规律即可。

编码

```
#include<bits/stdc++.h>

using namespace std;

const int Base = 2011;
const int POW0 = 1;  //0次方的时候等于1
const int POW = 10000; //取余末尾四位数

int roundNum; //循环的周期
int k;   //测试数据数量
int vs[1000] = {POW0, Base};  //不同幂次下，后四位的值的缓存
string cs;  //幂次的字符串形式
//0次方为1,1次方为2011

int main() {
    int i = 2;
    while (true) {
        vs[i] = Base * vs[i - 1] % POW;
        //找到了循环周期
       if (vs[i] == POW0) {
            roundNum = i;
            break;
        }
        i++;
    }
    cin >> k;
    for (int i = 0; i < k; i++) {
        cin >> cs;
        int len = cs.size();
        int n = 0;   //当前测试用例幂次
       for (int j = 0; j < len; j++) {
            //将指数幂根据周期取余
          n = (n * 10 + cs[j] - '0') % roundNum;
        }
        //从缓存中读取数据
       cout << vs[n] << endl;
    }
    return 0;
}
```