B 盗墓笔记之秦岭神树——Officially

题目描述

在天真和老痒还被困在棺材阵中时，Thor 却阴差阳错地走了出去，走了另一条不为人知的诡异捷径，率先来到了巨树。好奇的他，独自一人，向上爬去，很快便遭遇了大波的螭蛊。经过观察，他发现，这棵树的螭蛊分布如下所示：

第一层：1
第二层：1 x
第三层：1 2x x^2
第四层：1 3x 3x^2 x^3
第五层：1 4x 6x^2 4x^3 x^4
.
.
.

现在 Thor 想知道第 n 层有多少只螭蛊，但是我们已经知道了，数学这个东西从来就不是 Thor 擅长的，因此这个任务就交给了你。

输入

多组测试数据，每组测试数据为一行，包含两个整数 n 与 x（1≤n≤1000，1≤x≤1000），含义见题目描述。

输出

对于每组测试数据，输出一个整数，为第 n 层螭蛊的个数，结果对 100007 取模。
输入以两个 0 结尾

输入样例

1 100
2 4
3 2
0 0

输出样例

1
5
9

解题分析

考虑到大家的意见，本题没有过分地卡时间复杂度，直接使用杨辉三角进行递推计算也不会超时。因此衍生出两种算法，第一种，运用杨辉三角中某一点的系数等于其上方左右两元素和迭代得到第 n 行的系数,然后进行累加计算即可。（题目中给出的图可以看出其系数是一个杨辉三角）

然而注意这里的数据范围十分巨大，直接计算后取模会严重超出 int 范围，怎么做？前几次上机中，曾经出现过过程中取余的计算方法，可以参考当时的解题报告。

此外，在这里说一下正统解法：

杨辉三角实际上有另外一个性质，它的每一行，实际上是(1+x)^(n-1)的二项式展开得到的多项式的拆分形式，因此本题实际上是一个求幂问题，这样以来，本题就完全退化成了之前上机中计算大数模的题目。

参考代码：

```
#include<iostream>
#include<fstream>
#define m 100007
using namespace std;
int main()
{
     int n,x,sum;
     while(1){
         sum=1;
         cin>>n>>x;
         if(n==0&&x==0)break;
         for(int i=0;i<n-1;i++)
             sum=(sum*(x+1))%m;  //过程中求模
         cout<<sum<<endl;
     }
     return 0;
}
```

此外这里着重强调一点，cmath 库中绝大多数的库函数无论是参数还是返回值，都是浮点型。编程时出现编译错误无非两种情况：①传入了整形的参数②求模计算时出现了浮点数的返回值。对于上述两种情况，需要做的只是进行 static_cast 的强制类型转换即可。