用1,2,...,n表示n个盘子，称为1号盘，2号盘,...。号数大盘子就大。经典的汉诺塔问 题经常作为一个递归的经典例题存在。可能有人并不知道汉诺塔问题的典故。汉诺塔来源于 印度传说的一个故事，上帝创造世界时作了三根金刚石柱子，在一根柱子上从下往上按大小 顺序摞着64片黄金圆盘。上帝命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱 子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一回只能移动一个圆盘。我们   
知道最少需要移动2^64-1次.在移动过程中发现，有的圆盘移动次数多，有的少。告之盘子总数和盘号，计算该盘子的移动次数.

**Input**

包含多组数据，首先输入T,表示有T组数据.每个数据一行，是盘子的数目N(1<=N<=60)和盘   
号k(1<=k<=N)。

**Output**

对于每组数据，输出一个数，到达目标时k号盘需要的最少移动数。

**Sample Input**

2

60 1

3 1

**Sample Output**

576460752303423488

4

另一种做法

在演草纸上画一画就能找到规律。若是求移动盘子的总次数，那么有这样的一个关系：f[n] = 2 \* f[n-1] + 1;可以看出来，最下面的盘子只需要移动一次，倒数第二个盘子需要移动2次，通过这个式子可以看出递推关系为2倍。

#include <stdio.h>

#include <math.h>

#include<iostream>

using namespace std;

\_\_int64 f[66] = {1};

int main()

{

// freopen("input.txt","r",stdin);

int n, a, i, t;

for(i = 1; i <65; ++i)

f[i] = f[i-1]\*2;

cin>>t;

while(t--){

cin>>n>>a;

int x=n-a;

cout<<f[x]<<endl;

}

return 0;

}

#include <stdio.h>

#include <math.h>

#include<iostream>

using namespace std;

\_\_int64 pow(int x)

{

\_\_int64 sum(1);

for(int i=1;i<=x;i++)

sum\*=2;

return sum;

}

int main()

{

// freopen("input.txt","r",stdin);

int n,k,t;

\_\_int64 x;

cin>>t;

while(t--)

{

cin>>n>>k;

x=pow(n-k);

cout<<x<<endl;

}

return 0;

}

肯定是关于n，k的函数