Squares and rectangles fascinated the famous Dutch painter Piet Mondriaan. One night, after producing the drawings in his 'toilet series' (where he had to use his toilet paper to draw on, for all of his paper was filled with squares and rectangles), he dreamt of filling a large rectangle with small rectangles of width 2 and height 1 in varying ways.

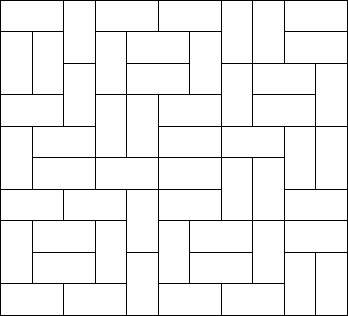
https://odzkskevi.qnssl.com/91cd2a849209448a71d50fa577e5d5ed?v=1502368465

Expert as he was in this material, he saw at a glance that he'll need a computer to calculate the number of ways to fill the large rectangle whose dimensions were integer values, as well. Help him, so that his dream won't turn into a nightmare!

Input

The input contains several test cases. Each test case is made up of two integer numbers: the height h and the width w of the large rectangle. Input is terminated by h=w=0. Otherwise, 1<=h,w<=11.

Output

For each test case, output the number of different ways the given rectangle can be filled with small rectangles of size 2 times 1. Assume the given large rectangle is oriented, i.e. count symmetrical tilings multiple times.

Sample Input

1 2

1 3

1 4

2 2

2 3

2 4

2 11

4 11

0 0

Sample Output

1

0

1

2

3

5

144

51205

经典状压dp

#include <iostream>

#include <string.h>

using namespace std;

int n,m;

long long dp[21][1<<12];

void dfs(int j,int i,int state,int nex)//状态的第j位数

{//i是每一行

if (j==m)//到头了

{

dp[i+1][nex]+=dp[i][state];

//dp[i+1][nex]%=mod;

return ;

}

//这个位置不能填

if (((1<<j)&state)>0)

dfs(j+1,i,state,nex);

else

{

//竖着

dfs(j+1,i,state,nex|(1<<j));//竖着放，这一行只占一个空而，但会影响下一行

//横着

if (j+1<m && ((1<<(j+1))&state)==0)//横着放,这一行占两个空,但不会对下一行产生影响

dfs(j+2,i,state,nex);

}

}

int main()

{

while (cin>>n>>m)

{

if(n==0 && m==0)return 0;

memset(dp,0,sizeof(dp));

dp[0][0]=1;

for (int i=0;i<n;i++)

for (int state=0;state<(1<<m);state++) //2^n个状态

if (dp[i][state]>0)

dfs(0,i,state,0);

cout<<dp[n][0]<<endl;

}

}