**Problem F**

**Optimal Transformation Cost**

Input File: pf.in

Time Limit: 3 seconds

A mathematician, Professor Lee, is now studying a transformation scheme in Coding Theory. There are 2*n* *n*-bit binary strings *S* = {*b*n*b*n-1…*b*i…*b*2*b*1|*b*i ∈ {0, 1} for 1 <= i <= n}. Two strings, can be transformed each other if and only if one is *b*n*b*n-1…*b*k+10*b*k-1*b*k-2…*b*1 and the other is *b*n*b*n-1…*b*k+11*b*k-1*b*k-2…*b*1, where *i* = 0, 1 and 1<= *k* <= *n* (i.e., their binary strings differ in a one-bit position only). We use *x*↔*y* to denote the transformation between two strings *x* and *y*, and use cost(*x*, *y*) to denote the cost of the transformation *x*↔*y*. To make the problem much easier, we assume the cost of each transformation is a constant *c*.

Professor Lee aims at finding a sequence of transformations *T*(*S*)=〈s1↔s2↔s3↔…sm-1↔sm(=s1)〉among S such that the following two conditions hold:

1. Every possible transformation is contained at least once by *T*(*S*).
2. The transformation cost *T*(*S*), defined by sigma(cost(*s*i, *s*i+1)), is as smallest as possible.

The minimum transformation cost of *T*(*S*) is called the optimal transformation cost, denoted by *cost*(*T*(*S*)). For example, consider *S* = {000, 001, 010, 011, 100, 101, 110, 111} and assume that *c* = 1. Then, *T*(*S*) =〈000↔001↔011↔010↔000↔100↔101↔001↔101↔111↔011↔111↔110↔010↔110↔100↔000〉and *cost*(*T*(*S*)) = 16. Note that *T*(*S*) may not be unique, but *cost*(*T*(*S*)) is unique.

Given a positive integer n and the cost of each transformation c, you task is to write a computer program to calculate the optimal cost *cost*(*T*(*S*))

**Technical Specification**

* 2 <= n <= 20
* 1 <= c <= 100

The first line of the input file contains an integer, denoting the number of test cases to follow. For each test case, one line contains two integers n and c separated by a space.

For each test case, output the optimal cost.