HDOJ 2276 – Kiki & Little Kiki 2

——by A Code Rabbit

Description

有 n 盏灯，编号从 1 到 n。
他们绕成一圈，也就是说，1 号灯的左边是 n 号灯。
如果在第 t 秒的时候，某盏灯左边的灯是亮着的，那么就在第 t+1 秒的时候改变这盏灯的状态。
输入 m 和初始灯的状态。
输出 m 秒后，所有灯的状态。

Types

Maths :: Matrix

Analysis

矩阵乘法和快速幂。
这题的关键就是构造矩阵。
每盏灯的状态下一秒的状态，和他当前的状态，还有左边的灯的状态有关系。
所以我们可以构造出这么一个矩阵：

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}$$

然后就好办了，矩阵乘法快速幂一下。
但是要注意，由于是灯的开关问题，
要 mod 2 运算，也可以用位运算，效率会高一点。
mod 2 用得不好有可能会 TLE。
（话说刚开始题目看错，对矩阵元素开了 long long，后来忘记改回来，结果 TLE 了。）
（顺便吐槽一下题目的标题，是鸡鸡和小鸡鸡 2 吗？）

Solution

```
// HDOJ 2276
// Kiki & Little Kiki 2
// by A Code Rabbit
```

```
#include <cstdio>
#include <cstring>

const int LIMITS = 102;

struct Matrix {
    int element[LIMITS][LIMITS];
};

int n;
int m;
char t[LIMITS];

Matrix mat_unit;
Matrix mat_one;
Matrix mat_ans;

void INIT();

Matrix QuickPower(Matrix mat_result, Matrix mat_one, int index);
Matrix Multiply(Matrix mat_a, Matrix mat_b);

int main() {
    while (scanf("%d", &m) != EOF) {
        getchar();
        // Inputs.
        gets(t);
        n = strlen(t);
        // Initialize.
        INIT();
        // QucikPower.
        mat_ans = QuickPower(mat_unit, mat_one, m);
        // Compete and outputs.
        for (int i = 0; i < n; ++i) {
            int sum = 0;
            for (int j = 0; j < n; ++j) {
                sum ^= (t[j] - '0') & mat_ans.element[j][i];
            }
            printf("%d", sum);
        }
        printf("\n");
```

```
    }

    return 0;
}

void INIT() {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            mat_unit.element[i][j] = i == j ? 1 : 0;
        }
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if ((i + 1) % n == j || i == j) {
                mat_one.element[i][j] = 1;
            } else {
                mat_one.element[i][j] = 0;
            }
        }
    }

}

Matrix QuickPower(Matrix mat_result, Matrix mat_one, int index) {
    while (index) {
        if (index & 1) {
            mat_result = Multiply(mat_result, mat_one);
        }
        mat_one = Multiply(mat_one, mat_one);
        index >>= 1;
    }
    return mat_result;
}

Matrix Multiply(Matrix mat_a, Matrix mat_b) {
    Matrix mat_result;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            mat_result.element[i][j] = 0;
            for (int k = 0; k < n; ++k) {
                mat_result.element[i][j] ^= mat_a.element[i][k] & mat_b.element[k][j];
```

```
            }
        }
    }
    return mat_result;
}
```