HDOJ 2256 – Problem of Precision

——by A Code Rabbit

Description

求题中那个有根号，有幂，有下取整的变态公式的值。

Types

Maths :: Matrix

Analysis

这题首先要注意不能将一个实数取模，即使你把它拆成整数部分和小数部分。

要用数学方法来解这题，我是解释不来的，所以这里引用一下牛人的推导：

$$(\sqrt{2}+\sqrt{3})^{2n}=(5+2\sqrt{6})^{n}=x_n+y_n\sqrt{6}$$

$$\Rightarrow x_n+y_n\sqrt{6}=(x_{n-1}+y_{n-1}\sqrt{6})\cdot(5+2\sqrt{6})=5x_{n-1}+12y_{n-1}+(2x_{n-1}+5y_{n-1})\sqrt{6}$$

$$\Rightarrow x_n=5x_{n-1}+12y_{n-1},y_n=2x_{n-1}+5y_{n-1}.$$

$$\Rightarrow\begin{bmatrix}x_n\\y_n\end{bmatrix}=\begin{bmatrix}5&12\\2&5\end{bmatrix}.\begin{bmatrix}x_{n-1}\\y_{n-1}\end{bmatrix}$$

$$----------$$

$$(5-2\sqrt{6})^{n}=x_n-y_n\sqrt{6}=(0.101...)^{n}$$

$$\Rightarrow x_n+y_n\sqrt{6}=x_n+y_n\sqrt{6}+x_n-y_n\sqrt{6}-(x_n-y_n\sqrt{6})=2x_n-(0.101..)^{n}$$

$$\Rightarrow ans=2x_n-1$$

推倒~之后，我们就可以利用矩阵乘法去求 Xn，然后计算出梦寐以求的 ans 了。

Solution

```
// HDOJ 2256
// Problem of Precision
// by A Code Rabbit

#include <cstdio>
#include <cmath>

const int DIVISOR = 1024;
```

```
struct Matrix {
    int element[2][2];
};

int t;

int n;

Matrix mat_one;
Matrix mat_ans;

void INIT();
void Multiply(Matrix& mat_a, Matrix mat_b);

int main() {
    scanf("%d", &t);
    while (t--) {
        // Inputs.
        scanf("%d", &n);
        // INIT.
        INIT();
        // Quick Power.
        --n;
        /* Just need to compete to n - 1 */
        while (n) {
            if (n & 1) {
                Multiply(mat_ans, mat_one);
            }
            n >>= 1;
            Multiply(mat_one, mat_one);
            //printf("%d %d\n", mat_ans.element[0][0], mat_ans.element[0][1]);
            //printf("%d %d\n", mat_ans.element[1][0], mat_ans.element[1][1]);

        }
        // Compete.
        int x_n = (5 * mat_ans.element[0][0] + 2 * mat_ans.element[1][0]) % DIVISOR;
        printf("%d\n", (x_n * 2 - 1) % DIVISOR);
    }

    return 0;
}
```

```
void INIT() {
    mat_one.element[0][0] = 5;
    mat_one.element[0][1] = 2;
    mat_one.element[1][0] = 12;
    mat_one.element[1][1] = 5;
    mat_ans.element[0][0] = 1;
    mat_ans.element[0][1] = 0;
    mat_ans.element[1][0] = 0;
    mat_ans.element[1][1] = 1;
}

void Multiply(Matrix& mat_a, Matrix mat_b) {
    Matrix mat_c;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            mat_c.element[i][j] = 0;
            for (int k = 0; k < 2; k++) {
                mat_c.element[i][j] += mat_a.element[i][k] * mat_b.element[k][j];
            }
            mat_c.element[i][j] %= DIVISOR;
        }
    }
    mat_a = mat_c;
}
```