



设  $x$  和  $y$  为当前项的前两项，从第三项开始，每四项与他们的关系是：

$$x + y$$

$$x + y - 1$$

$$x + y$$

$$x + y + 1$$

于是便可以构造以下矩阵：

$$(A_n, A_{n-1}, A_{n-2}, A_{n-3}, 1) =$$

$$(A_{n-1}, A_{n-2}, A_{n-3}, A_{n-4}, 1) \begin{pmatrix} 5 & 3 & 2 & 1 & 0 \\ 3 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

然后进行矩阵乘法和快速幂求解。

## Solution

### 1. 暴力求解

```
// HD0J 2604
// Queuing
// by A Code Rabbit

#include <iostream>
#include <string>

using namespace std;

int ans;
int n;

void Search(string str);

int main() {
    while (cin >> n) {
        ans = 0;
        Search("");
        cout << ans << endl;
    }
}
```

```
    }  
}  
  
void Search(string str) {  
    if (str.length() >= n) {  
        if (str.find("fmf") == string::npos &&  
            str.find("fff") == string::npos)  
        {  
            ++ans;  
        }  
        return;  
    }  
    Search(str + "f");  
    Search(str + "m");  
}
```

```
// HD0J 2604  
// Queuing  
// by A Code Rabbit
```

## 2. 递推+矩阵乘法+快速幂

```
#include <cstdio>
```

```
struct Matrix {  
    int element[5][5];  
};
```

```
int l, m;
```

```
const Matrix mat_init = {  
    5, 3, 2, 1, 0,  
    3, 2, 1, 1, 0,  
    0, 0, 0, 0, 0,  
    0, 0, 0, 0, 0,  
    1, 1, 1, 0, 1,  
};
```

```
const Matrix mat_unit = {  
    1, 0, 0, 0, 0,  
    0, 1, 0, 0, 0,  
    0, 0, 1, 0, 0,  
};
```

```
    0, 0, 0, 1, 0,
    0, 0, 0, 0, 1,
};

Matrix mat_one, mat_ans;

const int ans[] = {
    9, 6, 4, 2, 1,
};

void INIT();

void Multiply(Matrix& mat_a, Matrix mat_b);

int main() {
    while (scanf("%d%d", &l, &m) != EOF) {
        // INIT.
        INIT();
        // Quick Sort.
        int num = (l - 1) / 4;
        while (num) {
            if (num & 1) {
                Multiply(mat_ans, mat_one);
            }
            Multiply(mat_one, mat_one);
            num >>= 1;
        }
        // Compute and outputs.
        int ans_sum = 0;
        for (int i = 0; i < 5; ++i) {
            ans_sum += ans[i] * mat_ans.element[i][4 - ((l - 1) % 4 + 1)];
        }
        printf("%d\n", ans_sum % m);
    }

    return 0;
}

void INIT() {
    mat_one = mat_init;
    mat_ans = mat_unit;
}
```

```
}  
  
void Multiply(Matrix& mat_a, Matrix mat_b) {  
    Matrix mat_c;  
    for (int i = 0; i < 5; ++i) {  
        for (int j = 0; j < 5; ++j) {  
            mat_c.element[i][j] = 0;  
            for (int k = 0; k < 5; ++k) {  
                mat_c.element[i][j] += mat_a.element[i][k] * mat_b.element[k][j];  
            }  
            mat_c.element[i][j] %= m;  
        }  
    }  
    mat_a = mat_c;  
}
```