1327：黑白棋子的移动

题目描述

有2n个棋子（n≥4）排成一行，开始位置为白子全部在左边，黑子全部在右边，如下图为n=5的情形：

○○○○○●●●●●

移动棋子的规则是：每次必须同时移动相邻的两个棋子，颜色不限，可以左移也可以右移到空位上去，但不能调换两个棋子的左右位置。每次移动必须跳过若干个棋子（不能平移），要求最后能移成黑白相间的一行棋子。如n=5时，成为：

○●○●○●○●○●

任务：编程打印出移动过程。

输入

输入n。

输出

移动过程。

输入样例

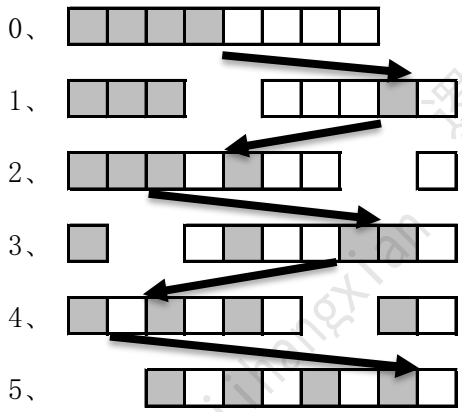
7

输出样例

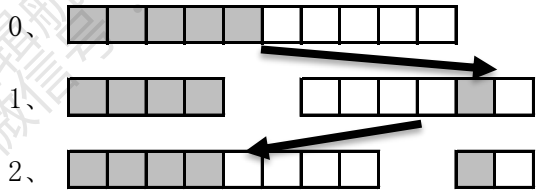
```
step 0:ooooooo*******--
step 1:oooooo--******o*
step 2:oooooo******--o*
step 3:ooooo--*****o*o*
step 4:ooooo*****--o*o*
step 5:oooo--****o*o*o*
step 6:oooo****--o*o*o*
step 7:ooo--***o*o*o*o*
step 8:ooo*o**--*o*o*o*
step 9:o--*o**oo*o*o*o*
step10:o*o*o*--o*o*o*o*
step11:--o*o*o*o*o*o*o*
```

解析

我们先来观察样例数据，发现当出现4黑4白的时候，似乎是最简模型。如下图所示：



经过上面的五步，我们就完成了移动。那么对于更多的黑白块的时候，我们该怎么办



通过两步移动，5黑5白就编程了4黑4白。同样的，对于6黑6白，和7黑7白，甚至更多的样

编码

```
#include<bits/stdc++.h>

const int N = 1001;

//https://blog.csdn.net/qq_42815188/article/details/101111252

using namespace std;
int n, cnt, space;    //space始终指向相邻两个空格的第一个
char a[N];

//打印函数
void print() {
    //保留两位宽度进行输出
   printf("step%2d:", cnt++);
    //打印全部的状态
   for (int i = 1; i <= 2 * n + 2; i++)
        printf("%c", a[i]);
    printf("\n");
}

//将以m作为起始索引的棋子移动到以space作为起始的空白处
void move(int m) {
    a[space] = a[m];
    a[space + 1] = a[m + 1];
    a[m] = a[m + 1] = '-';
    //记录新的空白
```

```
    space = m;
    print();
}

//递归移动函数
//m为棋子数的一半
void mv(int m) {
    if (m == 4)     //n==4相当于递归边界，按照固定步骤进行移动
    {
        move(4);
        move(8);
        move(2);
        move(7);
        move(1);
    } else     //n>4时，先移动两步达到n-1的状态，重复递归，直到n==4
    {
        move(m);
        move(2 * m - 1);
        mv(m - 1);
    }
}
int main() {
    scanf("%d", &n);
    //初始化白子、黑子、空位
    for (int i = 1; i <= n; i++) {
        a[i] = 'o';
    }
    for (int i = n + 1; i <= 2 * n; i++) {
        a[i] = '*';
    }
    for (int i = 2 * n + 1; i <= 2 * n + 2; i++) {
        a[i] = '-';
    }
    cnt = 0;
    //记录当前起始的空白索引
    space = 2 * n + 1;

    //打印初始化状态
    print();
    mv(n);     //递归移动棋子
    return 0;
}
```