**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10710: Chinese Shuffle**

**解題者：陳彥鋒**

**解題日期：2014年4月10日**

**題意：**

 每行輸入一個整數N(54 ≤ N ≤ $2\*10^{9}$)，代表一共有N張牌，需判斷經過(N-1)次perfect shuffle之後，是否可以回到原本順序，若可以，則輸出N is a Jimmy-number；反之，則輸出N is not a Jimmy-number，當輸入-1時，代表結束。

**題意範例：**

54 🡺 54 is not a Jimmy-number

 101 🡺 101 is a Jimmy-number

 144 🡺 144 is a Jimmy-number

 -1

**解法：**

 perfect shuffle是將牌分兩堆後，互相將牌交錯排序，因此每一次在相同位置的牌在經過perfect shuffle後，會移動到同一個位置，如果有N張牌要洗，在經過x次perfect shuffle後，第一張牌的位置將在$2^{x}\%N$，而當第一張牌回到原位置時，亦為所有牌都回到原位置，因此若$2^{(N-1)}\%N=1$，則代表經過(N-1)次perfect shuffle後，所有牌都回到原位置。

**解法範例：**

以5張牌作示範

Original ：1 2 3 4 5

Shuffle 1：3 1 4 2 5 🡺 第一次洗完牌後，編號1的牌在第(2%5=2)位

Shuffle 2：4 3 2 1 5 🡺 第二次洗完牌後，編號1的牌在第(4%5=4)位

Shuffle 3：2 4 1 3 5 🡺 第三次洗完牌後，編號1的牌在第(8%5=3)位

Shuffle 4：1 2 3 4 5 🡺 第四次洗完牌後，編號1的牌在第(16%5=1)位

5張牌在經過4次perfect shuffle之後，編號1回到第1為，而所有牌也都回到原本位置，因此5是Jimmy-number。

**討論：**

(1).因為N值最大會是$2\*10^{9}$，但推出來的公式是$2^{(N-1)}\%N=1$，就算使用long int 也只能到$2^{63}$。

(2). (A\*B)%N = (A%N)\*(B%N)，我們可以利用此性質，將公式變成：

當P為：

1： $2^{1}\%N=2$

偶數：$2^{P}\%N=\{[2^{P/2}\%N]\*[2^{P/2}\%N]\}\%N$

奇數：$2^{P}\%N=2\*\{[2^{(P-1)/2}\%N]\*[2^{(P-1)/2}\%N]\}\%N$

P之起始值為N-1，如此一來即可利用recursive來得到洗完(N-1)次牌後，編號1的牌位置在哪裡。

**程式：**

#include <stdio.h>

long long int big\_mod(int B, int P, int M);

int main(){

 int N = 1, ans;

 while(N != -1){

 scanf("%d", &N);

 if(N != -1){

 if(big\_mod(2, N-1, N) == 1){

 printf("%d is a Jimmy-number\n", N);

 }

 else{

 printf("%d is not a Jimmy-number\n", N);

 }

 }

 }

 return 0;

}

long long int big\_mod(int B, int P, int M){

 long long int R;

 if(P>1){

 if(P%2){

 R = big\_mod(B, (P-1)/2, M);

 return (B\*((R\*R)%M))%M;

 }

 else{

 R = big\_mod(B, P/2, M);

 return (R\*R)%M;

 }

 }

 else{

 return B%M;

 }

}