[A - Swap Bits](https://vjudge.net/problem/HackerRank-si-swap-bits" \t "_blank)

Given a number, swap the adjacent bits in the binary representation of the number, and print the new number formed after swapping.

**Input Format**

First line of input contains T - number of test cases. Each of the next T lines contains a number N.

**Constraints**

1 <= T <= 100000  
0 <= N <= 109

**Output Format**

For each test case, print the new integer formed after swapping adjacent bits, separated by new line.

**Sample Input**

4  
10  
7  
43  
100

**Sample Output**

5  
11  
23  
152

**Explanation**

**Test Case 1**

Binary Representation of 10: 000...1010  
After swapping adjacent bits: 000...0101 (5)

**Test Case 2**

Binary Representation of 7: 000...0111  
After swapping adjacent bits: 000...1011 (11)

**APPROACH 1:**

#include <iostream>

using namespace *std*;

bool isKthBitSetV1(int n, int k) {

return (n & (1LL << k)) != 0;

}

void setKthBit(int& n, int k) {

n = n | (1LL << k);

}

int main()

{

int t; *cin* >> t;

while (t--)

{

int n; *cin* >> n;

int newNum = 0;

for (auto i = 0; i < 32; i++)

{

bool preBit = isKthBitSetV1(n, i);

if (preBit)

setKthBit(newNum, i + 1);

i++;

bool curBit = isKthBitSetV1(n, i);

if (curBit)

setKthBit(newNum, i - 1);

}

*cout* << newNum << "\n";

}

return 0;

}

**APPROACH 2:**

N = 0101 1101

We finally want answer as

N = 1010 1110

So if we notice carefully, 0th bit and 1st bit are swapped. 2nd and 3rd bit are swapped. and so on.

In nutshell, 0th bit LEFT SHIFTS and 1st BIT RIGHT SHIFTS.

What if we had a way to just EXTRACT even bits of N (0th, 2nd, 4th, 6th)? leaving odd bits as 0s.

N = 0101 1101

Only EVEN BITS

EB = EVEN BITS OF N = 0101 0101

Only ODD BITS

OB = ODD BITS OF N = 0000 1000

Just LEFT SHIFT even BITS and OR with RIGHT SHIFT of ODD BITS

ans = (EB << 1) | (OB >> 1)

**CODE:**

#include <iostream>

using namespace *std*;

int main()

{

int t; *cin* >> t;

while (t--)

{

int n; *cin* >> n;

int newNum = 0;

int evenBits = 0;

int evenBitMask = 0XAAAAAAAA;

int oddBits = 0;

int oddBitsMask = 0X55555555;

evenBits = evenBitMask & n;

oddBits = oddBitsMask & n;

newNum = (evenBits >> 1) | (oddBits << 1);

*cout* << newNum << "\n";

}

return 0;

}