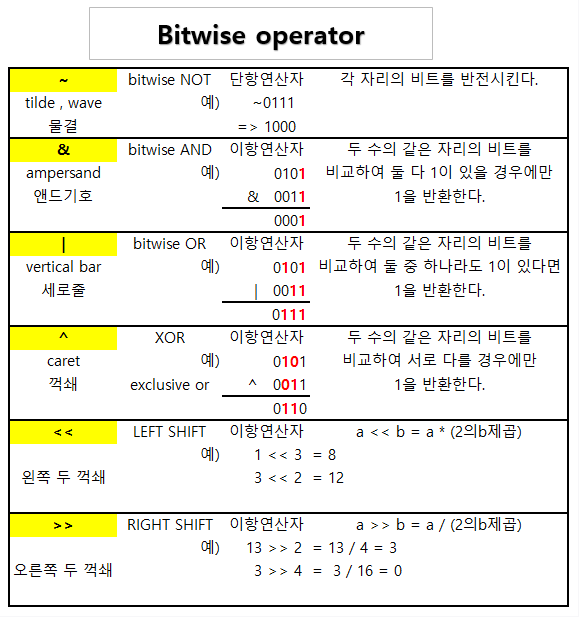
비트연산

비트연산(Bitwise Operation)은 한 개 또는 두 개의 이진수에 대하여 비트단위로 적용되는 연산이다.

다음은 6가지 비트 연산자에 대한 설명이다.



위 6개 비트 연산자를 사용하여 아래 질문에 답하시오.

질문에 사용된 모든 변수는 음이 아닌 정수이다.

문항별 사용되는 비트연산자는 1개를 초과할 수 있다.

(1) a가 홀수인지 if문과 비트 연산자를 이용하여 판별하고자 한다. ?자리에 들어갈 비트연산자​는?

if(a ? 1){} /// c, c++인 경우

if((a ?1) == 1 )) {} /// java의 경우

(2) a가 짝수인지 if문과 비트 연산자를 이용하여 판별하고자 한다. ?자리에 들어갈 비트​연산자​는?​

if(?a ? 1) {} /// c, c++인 경우​

if((?a ?1) == 1 )) {} /// java의 경우​

(3) a와 2의 k제곱을 곱한 결과(결과는 int 범위를 넘지 않는다.)를 구하고자 한다.​ ?자리에 들어갈 비트​연산자​는?​​

result = a ? k;

(4) a를 2의 k제곱으로 나눈 몫 p와 나머지 r 을 구하고자 한다. ?자리에 들어갈 비트​연산자​들은?​​​

p = a ? k, r = a ? ((1 ? k) - 1);

(5) 다음은 a와 b의 값을 서로 바꾸는 코드이다. ?자리에 들어갈 비트​연산자들은​?​​​

a = a ? b, b = a ? b, a = a ? b;

(6) a의 k번째(2의 k제곱 ​자리) 비트가 0인지 1인지 알아보고자 한다. ?자리에 들어갈 비트​연산자​들은​?​​​​

result = (a ? k) ? 1;

(7) a의 k번째(2의 k제곱 ​자리) 비트만 0으로 바꾸고자 한다. ?자리에 들어갈 비트​연산자​들은​?​​​​

a = a ? (?(1 ? k));

(8) a의 k번째(2의 k제곱 ​자리) 비트만 0이면 1로, 1이면 0으로 바꾸고자 한다. ?자리에 들어갈 비트​연산자​들은​?​​​​

a = a ? (1 ? k);

(9) a가 2의 제곱수인지 판별하고자 한다. ?자리에 들어갈 비트​연산자들은​?​​​​​

result = a ? (a - 1);

if( a > 0 && result==0) printf("%d is the power of 2.\n", a);

(10) a와 b의 같은 자리 비트를 비교한 결과 서로 다른 비트가 1개 이하인지 알아보고자 한다.

?자리에 들어갈 비트​연산자​들은​?​​​​​

bit = a ? b; result = bit ? (bit -1);

if(result==0) printf("%d and %d differ by below 1bit.\n", a, b); ​

위 질문에 대하여 아래 코드의 op[0]에서 0​위치에 적절한 비트 연산자 번호를 작성하여 제출한다.

#include <stdio.h>

char op[7][4] = {"?", "~", "&", "|", "^", "<<", ">>"};

int main(){

printf("%s\n", op[0]); /// (1)

printf("%s%s\n", op[0], op[0]); /// (2)

printf("%s\n", op[0]); /// (3)

printf("%s%s%s\n", op[0], op[0], op[0]); /// (4)

printf("%s%s%s\n", op[0], op[0], op[0]); /// (5)

printf("%s%s\n", op[0], op[0]); /// (6)

printf("%s%s%s\n", op[0], op[0], op[0]); /// (7)

printf("%s%s\n", op[0], op[0]); /// (8)

printf("%s\n", op[0]); /// (9)

printf("%s%s\n", op[0], op[0]); /// (10)

return 0;

출력형식

만약 모든 답이 ?라면 출력예와 같이 출력된다.

출력 예

?

??

?

???

???

??

???

??

?

??

**엔디안**

Lilliput의 사람들은 삶은 계란을 먹을 때 계란의 넓은 쪽부터 깨야 한다는 Big Endian파와

좁은 쪽부터 깨야 한다는 Little Endian 파로 나뉘어서 싸우고 있었다.

이 싸움은 삶은 계란 뿐만 아니라 생활 속의 다른 분야에까지 확대되었는데,

그 중 대표적인 것으로 컴퓨터에서 사용하는 데이터의 저장 방법이다.

Big Endian 파에서는 숫자를 저장할 때 위쪽 바이트부터 먼저 저장하는 방식이 옳다고 주장하였으며,

Little Endian 파에서는 아래쪽 바이트부터 먼저 저장하는 방식이 옳다고 주장하였다.

예를 들어 32bit unsigned int 305,419,896 (0x12345678)을 Big Endian과 Little Endian으로 나타내면 다음과 같다.

00010010 00110100 01010110 01111000 (Big Endian)

01111000 01010110 00110100 00010010 (Little Endian)

이 들은 각자 자신이 주장하는 방식으로 컴퓨터를 만들어서 사용하였는데,

두 파의 사람들이 서로 데이터를 주고 받을 필요가 있을 경우 (항상 싸우기만 하는 것은 아니다.)

Big Endian으로 저장된 데이터를 그대로 Little Endian 방식으로 읽었을 경우,

또는 반대의 경우에는 이상한 값이 되어버리기 때문에 Endian의 변환이 필요해진다.

예를 들어 위 예제의 Big Endian의 값을 그대로 Little Endian 방식으로 읽었을 경우 2,018,915,346 (0x78563412)이 된다.

이 들은 자신이 Endian을 변환해서 데이터를 보내주는 것을 원하지 않았기 때문에

일단 데이터를 받은 후에 자신들이 사용하는 Endian에 맞춰서 변환해야 한다.

당신은 이들을 도와 Endian을 변환하지 않고 받은 데이터를 원래의 데이터로 복원해주는 프로그램을 작성해야 한다.

입력형식

입력은 하나의 숫자 (32bit unsigned int)로 이루어지며, 이는 Endian을 변환하지 않고 받아들인 데이터이다.

출력형식

입력에 대한 원래 데이터를 출력한다.

입력 예

2018915346

출력 예

305419896

**비밀번호**

KOI 보안회사에서는 자동으로 비밀번호를 만드는 시스템을 연구하고 있다. 주어진 하나의 양의 정수에 대하여 다음의 원칙에 따라 두 수를 만들어 비밀번호를 정하려고 한다.

하나의 주어진 양의 정수 A에 대하여 비밀번호를 위한 두 수를 만드는 방법은 다음과 같다.

1. A의 이진수 표현에서 나오는 1의 개수 x를 찾는다.

2. A보다 작은수 중에서 그 수의 이진수 표현에서 1의 개수가 x와 같고 A에 가장 가까운 수를 하나 찾는다.

3. A보다 큰수 중에서 그 수의 이진수 표현에서 1의 개수가 x와 같고 A에 가장 가까운 수를 하나 찾는다.

예를 들어, 주어진 수 A가 43이면, 이 수의 이진수 표현은 101011(2)이다. 이 이진수는 1의 개수가 4이다. 그러므로 43보다 작고 43에 가장 가까우며 이진수 표현에서 1의 개수가 4인 수는 39=100111(2)이다. 또한 43보다 크고 43에 가장 가까우며 이진수 표현에서 1의 개수가 4인 수는 45=101101(2)이다.

이 두 수를 찾아 출력하는 프로그램을 작성하시오.

<제약조건>

• 전체 테스트 데이터의 30%는 1≤A≤104

• 전체 테스트 데이터의 60%는 1≤A≤108

입력형식

입력의 첫 번째 줄에는 하나의 양의 정수 가 주어진다. 단 1≤A≤1018이다.

출력형식

주어진 수 보다 작은 수 중에서 이진수의 1의 개수가 같으며 가장 가까운 수와, 주어진 수 보다 큰 수 중에서 이진수의 1의 개수가 같으며 가장 가까운 수를 한 줄에 빈칸을 사이에 두고 출력한다.

만약 그러한 수가 존재하지 않으면 그 수에 대해서는 0을 출력한다.

입력 예

43

출력 예

39 45

입력 예

7

출력 예

0 11