$\overset{0}{\boxed{1}}$ $n=1$

$1 = 2^0$ ✓

$\boxed{0}$ ✓ $n=2$ $\boxed{n-2=0}$

$\boxed{0\,1}$ ✓

Reordered power of 2

$\boxed{1 <= n <= 10^9}$

ex 1 :- $n = 1$

1] $\overset{0}{1}$ ✓

2] $\boxed{1}$

3] 1 ? 2^n ? T : F

ex 2 :- 128

1] ~~ht~~ no leading zero

2] $\boxed{128}$ 182 821 281 ... $\boxed{3! = 6}$

3] 128 → 2^7 ? $\boxed{T}$

$$\begin{tikzcd}\boxed{n} \arrow[r] & \boxed{n!}\end{tikzcd}$$

$\boxed{10^9 !}$ → TLE

Brute force

```
Bool solve(int n) {
    // step 1 - shuffle (next_permutation)
    // next_permutation(begin(), end())
    Time complexity :- n . n! = n.n!
                      calls
    ~~int~~ string s = to_string(n);
    //shuffle
    do {
        //leading zero
        if(s[0] == '0')
            continue

        n = stoi(s);
        if(n>0) {
            // power of 2?
            return(n & (n-1)) ? true : false;
        }
    } while(next_permutation(s.begin(),
            s.end());
    return false;
}
```

"108"

0 1 2

s = "081"

"811"

811

n>0

n!

n.n!

shuffle $\boxed{int}$ x

3-1 Overflow in Signed & Unsigned Numbers

2^{-3}	2^{-2}	2^{-1}	2^{0}
1	0	0	0

$\rightarrow -8$

8 4 2 1   8   8

$\boxed{1}\boxed{00}$

```
  1 0 0 0  → 4bits
+ 1 0 0 0  → 4bits
 10 0 0 0  → 5bits
```

0 → n
1 → n

$2^{?}$

$2^{3}-1$ 111

int → 32 bits

$2^{8}-1$ $2^{2}-1$ 7   7

11

$2^{31}-1$ $2^{n-1}-1$ $\boxed{1\ 1\ 1\ 1\ 1\ 1}$ $\boxed{2^{2}-1}$

421   010   111   4

111   0

$\boxed{-1}$ 0 $\boxed{\ }$ -2^{31} ⟷ $2^{31}-1$ 1 $\boxed{2^{2}-1}$

11

100   2^{0} 8

01

$2^{1}-1$

-128   -127

1   1

~~Binary~~ Overflow
Integer

2^{-31} to $2^{+31}-1$

- 1 bit is reserved to identify if it is signed or unsigned

- why -1 in $2^{31}-1$?

  1 used bcoz 0 is also a positive number that's 1 is included.

- In negative number, 2's complement is used to represent number in binary format.

Reordered Power of 2

Famous
Page No.:
Date:

* No shuffle .

2^n n

$2^6 = 64$ 46

sorted $\boxed{46 \quad = \quad 2 \quad 46}$ = 44

2^0 1
1 (1) = ~~X~~ 2 126

2^1 10 (2) = ~~X~~ 2 126

2^2 100 (4) = ~~X~~ 2 126

2^3 1000 (8) = ~~X~~ 126

2^4 10000 (16) = 2 126

2^5 10000 (32) = 2 126 ✗

2^6 1000000 (64) = 126 ✗

2^7 10000000 (128) = 126 ✗

(<<)

O(1)

Bitwise op

Left shift op
(<<)

* Syntax:-
(left op) (right op)
① First op << second op

* Exi-

1 byte char var = 3

binary = 0000 0011

var << 1

left shift by 1 pos

0 0 0 0 0 0 1 1

truncate

trailing pos filled ~0

0 0 0 0 0 0 1 1 0 = 6

<< operator

$\boxed{\text{<< is equivalent to multiplication by } 2^{\text{right operand}}}$

Ex:- var = 3;

var << 1 O/P:- 6 $[3 \times 2^1]$

Ex:- var << 4

3 << 4 O/P:- 48 $[3 \times 2^4]$
3×16

$\boxed{\begin{array}{l} 1 << 0 \\ 1 << 1 \end{array}}$

$1 \times 2^0 = 1 \times 1 = \boxed{1}$

$1 \times 2^1 = 1 \times 2 = \boxed{2}$