# Bit Manipulation

## Bitwise operators and their application

**Note: Time complexity of all the bitwise operator are approx O(1)**

& => bitwise AND

| => bitwise OR

^ => bitwise XOR

>> => bitwise Right shift

<< => bitwise Left shift

~ => bitwise NOT

**Note that & (bitwise AND) is different from && (logical AND) , | (bitwise OR) is different from || (logical OR) and ~ (bitwise NOT) is different from ! (logical NOT)**

**Rules->**

**AND**

0&0=0

0&1=0

1&0=0

1&1=1

**OR**

0&0=0

0&1=1

1&0=1

1&1=1

**XOR**

0&0=0

0&1=1

1&0=1

1&1=0

**Properties of XOR:**

a^a=0

0^a=a

**NOT**

~0=1

~1=0

| **int a=5 (101), b=7 (111); cout << a&b ; // (101) & (111)=(101)=5 cout << a|b ; // (101) | (111)=(111)=7 cout << a^b ; // (101) ^ (111)=(010)=2 cout << ~a ; // ~(101)=(010)=2** |
| --- |

**Left shift operator(<<)**

a -> 0000001110

a<<1 -> 000011100

**Right shift operator(>>)**

a -> 0000001110

a>>1 -> 0000000111

E.g int a=5;

a=a<<1; // then a? = 10 (2\*a);

a=5;

a=a>>1; // then a? = 2 (a/2) e.g (5/2)=2;

**Sol**

a=5(00101);

a<<1 -> (01010) = 10;

**Q.) You are given a number and find the value of (a<<b)?**

Ans: a\*(2b)

**Q.) You are given a number and find the value of (a>>b)?**

Ans: a/(2b)

**Why? 2^3 2^2 2^1 2^0 (2^2+2^0=5)**

**a=5 => 0 1 0 1**

**a<<1 => 1 0 1 0**

**(2^3+2^1)=2(2^2+2^0)=2\*a;**

**Thus, (1<<n) is equivalent to 2n**

**Q.) You are given an array of N numbers in which all the numbers are repeated twice except one number which is present exactly once then find out that number?**

Link: <https://www.hackerrank.com/challenges/lonely-integer/problem>

**e.g-> {2,3,4,4,2} so Answer=3;**

**Hint-> use XOR property**

**a^a=0;**

**a=5 => (101) then a^a (101)^(101) = (000)=0;**

**If we use xor of all the numbers present in the array i.e**

**2^3^4^4^2 = 3**

**If And-> (1&2&4&4&2)=>(1&2&3&4) No Need.**

| int main() { int n; cin>>n; vector<int> arr(n); for(int i=0;i<n;i++) cin>>arr[i]; int ans = 0; for(int i=0;i<n;i++){ ans=ans^arr[i]; } cout<<ans; } |
| --- |

**Q.) You have to check whether the given number is odd or even but you are not allowed to use % operator then how do you do that?**

a=5-> (0101)

b=10-> (1010)

a&1=>(0101)&(0001)=(0001)=1

b&1=>(1010)&(0001)=(0000)=0

| **if((a&1)==1){ cout<<"ODD"<<endl;  } else { cout<<"EVEN"<<endl; }** |
| --- |

**Q.) How to check whether ith bit (from right) is 1 or 0 for the given input number n.**

001010101010

….43210 (index)

0->0

1->1

(1<<i) -> 000000010000000

i

**Answer:**

| if(n&(1<<i)) {  cout<<"i-th bit is set"<<endl; } else {  cout<<"i-th bit is not set"<<endl; } |
| --- |

**Q.) How to set the ith bit to 1 for the given input number n;**

Ans:

n = n|(1<<i);

**Q.) How to set the ith bit to 0 for the given input number n;**

Ans:

n=n&(~(1<<i))

**a-> 0000010101010100000**

**i**

**(1<<i) 0000000010000000000**

**~(1<<i) 111111111011111111111**

**Q.) How to calculate the number of setbits(1) in the given number.**

**0<=n<=2^63-1**

**Sol:-**

| long long n; // 0<=n<=2^63-1  cin>>n;  long long ans=0;  // ans stores the number of set bits  for(int i=0;i<64;i++){  // 1<<i -> int  // 1LL<<i -> long long int  if(n&(1LL<<i)){  ans++;  }  }  cout<<ans; |
| --- |

**Q.) How to swap two numbers using the XOR operator.**

**Sol:** (x^y)^y -> x

(x^y)^x -> y

x = x^y; // x = x^y,y=y

y = x^y; // y = (x^y)^y -> x // x = x^y,y = x

x = x^y; // x^y^x = y // x=y,y=x

**Q.) How to generate all possible non empty subsequences of the given string.**

**e.g-> (abc) has following subsequences:**

**a,b,c,ab,bc,ac,abc**

Subsequence -> delete some elements from anywhere in the string and concatenate the remaining.

Eg abcdefgh -> del c,f,h -> abdeg

Substring -> delete some elements from the beginning and some from the end.

Eg abcdefgh -> del a,b from begin and g,h from end

-> cdef

**Sol :-**

Assume that length of string is n.

Represent any subsequence of this string as a binary number of length n.

Eg abc -> binary number of length 3

If binary digit is 1 -> then that char is present in this sequence.

Else it is deleted.

abc

110 -> ab 1 to 7 -> 001,010,011,100,101,110,111

-> c, b , bc, a ,ac ,ab , abc

101-> ac

N -> 1 to 2^n-1

| vector<string> seq;  // stores all non empty subsequences  string s;  cin>>s;  int n = s.length();  //n<=15  for(int i=1;i<(1<<n);i++){ // 1 to 2^n-1  string temp="";  for(int j=0;j<n;j++){  if(i&(1<<j)){ // if jth bit in i is set  temp+=s[j];  }  }  seq.push\_back(temp);  }  for(int i=0;i<seq.size();i++){  cout<<seq[i]<<" ";  } |
| --- |

**Q.** [**https://www.hackerrank.com/challenges/and-product/problem**](https://www.hackerrank.com/challenges/and-product/problem)

N queries -> a and b

Output a&(a+1)&(a+2)...b

12 and 15 -> 1100 and 1111 -> 1100 -> 12

14 15 -> 1110 and 1111 -> 1110 -> 14

1001 and 1101 -> 1000 -> 8

1001&1010&1011&1100&1101 -> 1000 - >8

01111 and 10000 -> 0

| **string x; for(int i=31;i>=0;i--){  if(a&(1<<i)){  x+='1';  else{  x+='0'; }** |
| --- |

| **while(a){  int x = a%2;  x+=('0'+x);  x/=2; } reverse(x.begin(),x.end());** |
| --- |

**Q.** [**https://www.hackerrank.com/challenges/sansa-and-xor/problem**](https://www.hackerrank.com/challenges/sansa-and-xor/problem)

**3 4 5**

**3**

**4**

**5**

**3,4**

**4,5**

**3,4,5**

Answer=> (3)^(4)^(5)^(3^4)^(4^5)^(3^4^5)

-> If occurence is even then don’t take it.

4 -> a^a^a^a = 0

-> If occurence is odd then include it in your answer.

5 -> a^a^a^a^a = a

| int sansaXor(vector<int> arr) {  int ans=0;  int n=arr.size();  for(int i=0;i<n;i++){  long occ=(i+1)\*(n-i);  if(occ&1) ans^=arr[i];  }  return ans; } |
| --- |