29-09-2023

(2105a41175) DAY3

BITWISE OPERATIONS:

Operations are:

|  |  |  |  |
| --- | --- | --- | --- |
| operations | c | python | java |
| Bitwise and | & | and | & |
| Bitwise or | | | or | | |
| Bitwise negotiation | ~ | ~ | ~ |
| Not | ! | not | ! |
| 1’s compliment |  |  |  |
| 2’s compliment |  |  |  |

PRIORITY OF BITWISE OPERATOR:

1)program:

#include<stdio.h>

int main()

{

printf("%d",10\*4/6+3-1%2);

}

OUTPUT: 8

2) program:

#include<stdio.h>

int main()

{

printf("%d",7+2&4+3&9);

}

OUTPUT:1

3)program:

#include<stdio.h>

int main()

{

printf("%d",10/3&4);

}

OUTPUT:1

EXCLUSIVE (XOR):

|  |  |  |
| --- | --- | --- |
| A | B | O/P |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

EX:10^5

10=>1 0 1 0

5=> 0 1 0 1

1 1 1 1

INCLUSIVE(OR):

|  |  |  |
| --- | --- | --- |
| A | B | O/P |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

BIT MANUPULATION TRICKS:

Xor:

Even 1’s:0

Odd 1’s:1

Xor of number itself is 0

Xor of number with 0 is number itself

1) 1^2^3

= 0001^0010^0011

= 0011^0011=>0

2) 4^6^5

= 0100^0110^0101

= 0010^0101=>0111(7)

RIGHT SHIFT:

1. 5>>1

0 1 0 1

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 0 |

2. 14>>2 14=>1 1 1 0 2=> 0 0 1 0

1 1 1 0

|  |  |  |  |
| --- | --- | --- | --- |
| X | 1 | 1 | 1 |

2^2 2^1 2^0=>4+2+1=7 =>0111

F0R 7

0 1 1 1

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 1 |

2^1 2^0=>2+1=3

LEFT SHIFT:

1. 5<<2 5=>0101 2=>0010

0 1 0 1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | X |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 0 |

2^3 0 2^1 0=>8+2=10

1.After creating an array write a program to find out the smallest missing positive integer

JAVA CODE:

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner in=new Scanner(System.in);

System.out.println("Enter array size:");

int n=in.nextInt();

int[] arr=new int[n];

System.out.println("Enter array elements:");

for(int i=0;i<n;i++){

arr[i]=in.nextInt();

}

checkWithCurr(arr); //Sorting O(n) time complexity

int ans=missing(arr);

System.out.println(ans);

}

static void checkWithCurr(int[] arr)

{

int i=0;

while(i<arr.length)

{

int correct=arr[i];

if(arr[i]>=arr.length)

{

i++;

}

else if(arr[i]<0){

i++;

}

else if(arr[i]==arr[correct])

{

i++;

}

else {

int temp=arr[i];

arr[i]=arr[correct];

arr[correct]=temp;

}

}

}

static int missing(int[] arr)

{

for(int i=0;i<arr.length;i++){

if(arr[i]!=i){

return i;

}

}

return arr.length;

}

}

PYTHON CODE:

b=input()

a=[int(x) for x in b.split(" ")]

print(a)

for i in range(len(a)):

if i not in a:

print(i)

break;

2. In the given array every integer occurs twice one integer occurs once

Java code:

def findSingle(ar,n):

res=ar[0]

for i in range(1,n):

res=res^ar[i]

return res

ar=[2,3,5,4,5,3,4,2,88]

print(findSingle(ar,len(ar)))

OUTPUT: 88

3.swap 2 numbers using XOR:

Python Program:

a=int(input("a="))

b=int(input("b="))

a=a^b

b=a^b

a=a^b

print("a=",a)

print("b=",b)

OUTPUT:

a=100

b=200

a= 200

b= 100

4. For the given number check whether the kth bit is set or not

Python Program:

n=int(input())

k=int(input())

a=n&(1<<(k-1))

if(a==0):

print("not a set")

else:

print("set")

OUTPUT: 10 3

Not a set

5. For the given number n find out XOR of all n numbers

1^2^3^4^5^6^7^8^9^10

|  |  |
| --- | --- |
| n | output |
| 1^2 | 1 |
| 1^2^3 | 3 |
| 1^2^3^4 | 0 |
| 1^2^3^4^5 | 4 |
| 1^2^3^4^5^6 | 1 |
| 1^2^3^4^5^6^7 | 7 |
| 1^2^3^4^5^6^7^8 | 0 |
| 1^2^3^4^5^6^7^8^9 | 8 |
| 1^2^3^4^5^6^7^8^9^10 | 1 |

Python program:

n=int(input("enter num"))

xor=0

if n%4==0:

print(n)

elif n%4==1:

print(1)

elif n%4==2:

print(n+1)

elif n%4==3:

print(0)

OUTPUT:

enter num:5

1

6. Find out the EXOR for all the given numbers in the range

Python Program:

from operator import xor

def findXOR(l, r):

ans = 0

for i in range(l,r+1):

ans = xor(ans,i)

return ans

l = 4; r = 8;

print(findXOR(l, r));

OUTPUT: 8

7. Check even or odd using bitwise:

Python program:

n=int(input())

if(n&1==0):

print(“even”)

elif(n&1==1):

print(“odd”)

OUTPUT: 8

Even

8. Fibonacci-ith term iterative

Python program:

n=int(input("enter the term"))

n1=0

n2=1

if(n<0):

print("not possible")

else:

for i in range(0,n-1):

n3=n1+n2

n1=n2

n2=n3

print(n2)

OUTPUT:

Enter the term:5

5