OPERAȚII PE BIȚI

1) paritate

x = 1001 x = 1000
↓ ↓
imp. par

x = 1001 & 1
0001
0001 ← arată că este imp

x & 1 == 0 => numărul este par

2) interschimbare

x = 1001 I. aux = x ^ y = 1001 ^ 0010 = 1011
y = 0010
aux

II. y = aux ^ y = x ^ y ^ y = 0 ^ x = x = 1011 ^ 0010 = 1001 = x

III. x = aux ^ x = x ^ y ^ x = 0 ^ y = y

3) $n = 2^k$? , K = ?

Ideea se bazează pe faptul că numerele de forma 2^k au urm. reprez.

000...100...0

deci, vom shifta la dreapta toate 0-urile de la final și dacă ne rămâne 1, înseamnă că $n = 2^k$.

Rez 1:

```
PYTHON:
n = int ( input ("n = "))
aux = n        // păstrăm n-ul
k = 0          // contorul
while aux & 1 == 0 :
        aux = aux >> 1
        k += 1
if aux == 1 :
        print (k)
else :
        print ("Nu este de forma 2^k")
```

Rez 2:

4) Numărul de biți nenuli din reprezentarea unui număr / Numărul de „1"

Ne vom folosi în mod repetitiv de n și n-1

n = 1001

n & (n-1) = 10011 & 10010 = 10010 (pas 1)

n = n & (n-1) =>

n & (n-1) = 10010 & 10001 = 10000 (pas 2)

... până când devine 0

PYTHON

```
n= int(input("n="))
ct=0
aux=n
while aux!=0:
    aux = aux & (aux-1)
    ct+=1
print(ct)
```

5) x, y ∈ ℕ

Calc. nr. biților din repr. internă a nr. x a căror val. trebuie comutată pentru a obține numărul y.

! operatorul xor este perfect pt. cazurile în care biții sunt diferiți

x = 1001

y = 0010

x ^ y = 1001 & 0010 = 1011 -> numărul de 1 din x^y = nr. de biți din repr. internă care trebuie comutați

PYTHON

```
x = int(input("x = "))
y = int(input("y = "))
ct=0
aux = x^y
/// urmează să calculăm nr. de 1 din repr. lui aux cu „>>"
while aux!=0:
    if aux%2==1
        ct+=1
    aux=aux>>1
print(ct)
```

6) avem un set de numere unde fiecare se repetă de 2 ori mai puțin unul

ex: 5 6 5 3 1 3 8 8 1

Aici, pt. a rezolva, ne bazăm pe proprietățile: $X \wedge X = 0$, $X \wedge 0 = X$

Astfel $5 \wedge 6 \wedge 5 \wedge 3 \wedge 1 \wedge 1 \wedge 3 \wedge 8 \wedge 8 \wedge 1 = 6$

PYTHON:

```
n= int ( input("n=") )
numar = 0
ct = 0   // numere citite
while ct != n:
        y = int ( input ( "y=") )
        aux = y ^ aux
        ct += 1
print ( aux )
```

7) lungimea maximă a unei secv. de biți egali cu 1 din reprez. unui nr.

- Soluție brută cu calculul lungimii și shiftare la dreapta
- soluție cu shiftare la stanga/dreapta și cu „&"

```
n= 0100111
n<<1 1001110
-------  „&"
0000110
```

am eliminat câte un „1" din fiecare secv., iar nr. de pași făcuți până n==0 = lgmax secv. de 1

PYTHON

```
n= int ( input ( "n= ") )
ct = 0
while n!=0 :
      n= n & (n <<1)
      ct += 1
print (ct)
```