Programarea Algoritmilor
– SEMINAR NR. 1 –
(grupa 131)

1. Să se interschimbe valorile a două variabile de tip întreg folosind operatorul ^ (XOR/sau exclusiv pe biți).

```
x = int(input("x = "))
y = int(input("y = "))
print("x =", x, ", y =", y)
x = x ^ y
y = x ^ y
x = x ^ y
print("x =", x, ", y =", y)
```

2. Să se verifice dacă un număr natural nenul x este de forma 2^k sau nu. În caz afirmativ să se afișeze valoarea k.

```
x = int(input("x = "))
if x & (x-1) == 0:
    k = 0
    aux = x
    while aux != 0:
        k = k + 1
        aux = aux >> 1
    print("Da\nk =", k-1)
else:
    print("Nu")
```

3. Să se determine în mod eficient numărul de biți nenuli din reprezentarea binară a unui număr natural.

```
# Solutia 1
x = int(input("x = "))
print("x =", bin(x))
k = 0
aux = x
while aux != 0:
    if aux & 1:   # daca aux e impar (are ultima cifra 1)
        k = k + 1
    aux = aux >> 1
print("Numărul", x, "are", k, "biți nenuli în reprezentarea binară.")

# Solutia 2
x = int(input("x = "))
print("x =", bin(x))
k = 0
aux = x
while aux != 0:
    k = k + 1
    aux = aux & (aux - 1)   # dispare cel mai din dreapta 1 al lui aux
print("Numărul", x, "are", k, "biți nenuli în reprezentarea binară.")
```

4. Fie x și y două numere naturale. Calculați numărul biților din reprezentarea binară internă a numărului x a căror valoare trebuie comutată pentru a obține numărul y.
Indicație de rezolvare: Se calculează numărul biților nenuli din x^y!

5. Se citește un șir format din numere naturale cu proprietatea că fiecare valoare distinctă apare de exact două ori în șir, mai puțin una care apare o singură dată. Să se afișeze valoarea care apare o singură dată în șir.

```
n = int(input("Numărul de valori: "))
x = k = 0
while k < n:
    v = int(input("Valoare: "))
    x = x ^ v
    k = k + 1
print("Numărul care apare o singură dată:", x)
```

6. Să se găsească lungimea maximă a unei secvențe de 1 din reprezentarea binară a unui număr natural dat.

```
# Solutia 1
x = int(input("x = "))
max = k = 0
aux = x
while aux >= 0:
    if aux & 1: # daca aux are ultima cifra 1
        k = k + 1
    else:
        if k > max:
            max = k
        if aux == 0:
            break
        k = 0
    aux = aux >> 1
print("x =", bin(x))
print("Lungimea maximă a unei secvențe de biți nenului este", max)

# Solutia 2
x = int(input("x = "))
k = 0
aux = x
while aux != 0:
    # lungimea fiecarei secvente de 1 se micsoreaza cu o unitate
    aux = aux & (aux << 1)
    k = k + 1
print("x =", bin(x))
print("Lungimea maximă a unei secvențe de biți nenului este", k)
```

7. Se citesc $n - 1$ numere naturale distincte din mulțimea $\{1, 2, \ldots, n\}$. Să se afișeze numărul lipsă. (**TEMĂ:** varianta în care lipsesc 2 numere)

```
# Solutia 1
n = int(input("n = "))
aparitii = [0] * n
k = 0
while k < n-1:
    v = int(input("v = "))
    aparitii[v-1] = 1
    k = k + 1
for i in range(len(aparitii)):
    if aparitii[i] == 0:
        print("Numărul lipsă =", i+1)
        break
```

```
# Solutia 2
n = int(input("n = "))
k = suma = 0
if n%2 == 0:
    suma = (n // 2) * (n + 1)
else:
    suma = ((n+1) // 2) * n
while k < n-1:
    v = int(input("v = "))
    suma = suma - v
    k = k + 1
print("Numărul lipsă =", suma)

# Solutia 3
n = int(input("n = "))
k = 0
x = 0
while k < n-1:
    v = int(input("v = "))
    x = x ^ v ^ (k + 1)
    k = k + 1
x = x ^ n
print("Numărul lipsă =", x)
```

TEMĂ: Se citesc $n-2$ numere naturale distincte din mulțimea $\{1, 2, \dots, n\}$. Să se afișeze numerele lipsă.

```
n = int(input("n = "))
v = []
x_xor_y = 0
for i in range(1, n-1):
    t = int(input("v[" + str(i) + "] = "))
    v.append(t)
    x_xor_y = x_xor_y ^ t
for i in range(1, n+1):
    x_xor_y = x_xor_y ^ i

bit = x_xor_y & ~(x_xor_y - 1)

x = 0
for t in v:
    if t & bit:
        x = x ^ t
for i in range(1, n+1):
    if i & bit:
        x = x ^ i

y = x_xor_y ^ x

print("Numerele lipsă: ", x, " și ", y)
```

8. Să se calculeze numărul obținut prin aplicarea operatorului XOR între toate elementele tuturor submulțimilor unei mulțimi nevide $A = \{a_1, a_2, \dots, a_n\} \subset \mathbb{N}$, mai puțin mulțimea vidă. De exemplu, pentru mulțimea $A = \{2,7,4\}$ trebuie afișată valoarea $v = (2)\text{^}(7)\text{^}(4)\text{^}(2\text{^}7)\text{^}(2\text{^}4)\text{^}(7\text{^}4)\text{^}(2\text{^}7\text{^}4) = 0$, unde am folosit parantezele pentru a evidenția submulțimile lui A.
Indicație de rezolvare: Demonstrați/argumentați faptul că orice valoare din mulțimea A va fi conținută într-un număr par de submulțimi, deci valoarea expresiei cerute va fi egală cu 0. Pentru $n = 1$ rămâne adevărată această proprietate a unui element al mulțimii A?