# Laborator 2 - Structuri de Date - 311CCa
# Recursivitate II

Bogdan Nutu, bogdan.nutu97@gmail.com

# Breviar

## 1. Divide et Impera

Divide et Impera se bazeaza pe principiul descompunerii problemei in doua sau mai multe subprobleme (mai usoare), care se rezolva, urmand ca solutia pentru problema initiala sa se obtina combinand solutiile subproblemelor.

De multe ori, subproblemele sunt de acelasi tip si pentru fiecare dintre acestea se poate aplica aceeasi tactica a descompunerii in alte subprobleme. Acest lucru se va repeta pana cand, in urma descompunerilor repetate, vom ajunge la probleme care admit rezolvare imediata.

## 2. Cautarea binara (Binary Search)

Problema cautarii binare presupune gasirea unui element x intr-un sir de elemente de acelasi tip ordonate crescator. In particular, sirul poate fi unul de numere intregi.

Pseudocod:

```
/* Inițial: low = 0, high = N - 1 */
int binary_search(int A[0..N-1], int x, int low, int high)
{
    if (high < low)
        return -1

    int mid = (low + high) / 2
    if (x < A[mid])
        return binary_search(A, value, low, mid - 1)
    else if (x > A[mid])
        return binary_search(A, value, mid + 1, high)
    else
        return mid
}
```

3. **Sortarea prin interclasare (MergeSort)**

Pseudocod:

```
int* merge_sort(int A[0..N-1])
{
    if (N <= 1)
        return A
    else {
        int mid = N/2
        int left[0..N/2-1] = A[0] ... A[mid-1]
        int right[0..N/2-1] = A[mid] ... A[N-1]

        merge_sort(left)
        merge_sort(right)

        return merge(left, right)
    }
}

int* merge(int A[0..M-1], int B[0..N-1])
{
    int C[0..N-1]

    /* Add elements from either A or B */
    while (i < M and j < N) {
        if (A[i] <= B[j])
            add A[i++] to C
        else
            add B[j++] to C
    }

    /* Add remaining elements from A */
    while (i < M)
        add A[i++] to C

    /* Add remaining elements from B */
    while (j < N)
        add B[j++] to C

    return C
}
```

# Exercitii

1. Sa da un numar intreg *n* si o cifra *c*. Sa se scrie o functie recursiva care calculeaza numarul obtinut prin eliminarea din *n* a tuturor aparitiilor cifrei *c*.
   Functia va avea antetul:
   **int eliminare_cifra (int n, int c);**
   Exemple:
   - eliminare_cifra (24599842, 2) = 459984;
   - eliminare_cifra (111, 1) = 0;
   - eliminare_cifra (1234, 5) = 1234;

2. Calculati logaritmul natural al unui numar real cu o precizie de 3 zecimale.
   **Hint**: Cautarea binara.
   Functia va avea antetul:
   **double my_ln (double x);**
   Exemple:
   - my_ln (2.5) = 0.916

3. Sa se implementeze algoritmul de sortare prin interclasare (MergeSort).

4. Sa se gaseasca indicele magic intr-un vector sortat de numere naturale, daca exista. Daca nu, se va returna -1.
   Indicele magic intr-un vector are proprietatea ***i = v[i]***.
   Exemple:
   - magic_index ([-1, 0, 1, 2, 4, 10]) = 4
   - magic_index ([1, 2, 3, 4, 6, 9, 11]) = -1

5. Imbunatatiti complexitatea functiei recursive de la problema 2 din laboratorul precedent folosind memorizare.