**Laborator 16**

**În REPORT.txt adăugați output-ul versiunii finale a programului. Dacă o parte din program nu e implementată, nu funcționează, face ca programul să dea seg fault atunci puteți comenta unele linii din main și să folosiți aceea afișare.**

**Exerciții**

1. (**mergeSort.c**) Implementați operația de merge din algoritmul de merge sort. **merge()**
2. (**mergeSortRec.c**) Implementați algoritmul [Merge Sort](https://en.wikipedia.org/wiki/Merge_sort) în varianta recursivă (ar trebui să apeleze funcția implementată la punctul anterior). **mergeSortRec()**
3. (**mergeSort.c**) Implementați algoritmul [Merge Sort](https://en.wikipedia.org/wiki/Merge_sort) în varianta iterativă (se pornește de la vectori de un element și se merge în sus, echivalent cu ieșirea din recursivitate). **mergeSort()**
4. (**quickSort.c**) Folosind funcția quicksort din librăria standard C ordonați un vector de numere descrescător **sortIntDesc()**, un șir de caractere alfabetic **sortChar()** și o structură de studenți descrescător (după note) **sortStudents()**.
5. (**myQuickSort.c**) Implementați algoritmul [Quick Sort](https://en.wikipedia.org/wiki/Quicksort). **myquickSort()**

**Exercițiile de la 1 la 5** sunt **obligatorii**. Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

**Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:**

1. (**radixSort.c**) Implementați algoritmul de sortare [Radix Sort](https://en.wikipedia.org/wiki/Radix_sort). **radixSort()**

**Exemplu afișare:**