Deadline 14 Martie 23:59 (atunci trebuie sa aveti ultimul update pe GitHub!)
Opțiunea 1:

- Implementați următorii 3 algoritmi de sortare:
    - RadixSort (ideal in mai multe baze inclusiv 2^16)
    - MergeSort
    - ShellSort
    + Inca 2 la algere dintre care unul in O(n logn) sau bazat pe numarare…

- Comparati timpii de rulare pe mai multe teste cu **numere naturale**, între cei 5 algoritmi dar și cu timpul de rulare al algoritmului de sortare nativ al limbajul de programare ales.
- Ideal este sa veniți cu o prezentare a acestor comparații (chiar cu niște slide-uri, rapoarte.)
- Puneți tot proiectul pe GitHub.
- Recomandare format fisier teste
    - Pe prima linie numărul de teste apoi pentru fiecare test cate numere trebuie sortate și care este cel mai mare număr.
        - T = 8
        - N = 1000 Max = 1000000
        - N = 10000 Max = 1000
        - ……
- Exemplu cod
    - For (test în tests) {
        - Generate numbers
        - Print (N = … Max =)
        - For sort în sorts
            - Timp_start = time()
            - Sort(v….)
            - Timp_end = time()
            - Print (“sortname timp_end-timp_stat test_Sort(v)”
- Unde test_sort(v) este o funcție care verifica ca algoritmul de sortare a sortat corect.
- Algoritmi trebuie testați pe diverse teste cu N = 1000, 10^6, 10^8, max 10^3, 10^?, 10^?. Algoritmii nu trebuie sa se blocheze sau sa dea segmentation fault indiferent de test, dar se pot opri și să spună ca nu pot sorta.

Optiunea 2:

- Același lucru de mai sus cu alți algoritmi … dacă algoritmii sunt complexi puteți implementa mai puțini
    - IntroSort sau TeamSort sunt mai complicati
    - InsertionSort nu este mai complicat

Observatii:

- ❏ Sortarile trebuie implementate cat de cat eficient de exemplu dacă RadixSortul vostru este mult mai încet ca QSort-ul pentru numere naturale mai mici ca 10^6 atunci ceva nu este bine la implementarea voastra…
- ❏ Testele alese ar trebui să evidențieze în ce cazuri anumiți algoritmi sunt mai buni ca alții
- ❏ La **Q-Sort** este de preferat sa alegeti pivotul ca fiind mediana din 3 sau 5 sau mediana medianelor, sau chiar mai bine sa alegeti pivotul în 2 moduri diferite și să arătați cum influențează timpul de rulare acest lucru.

- ❏ La RadixSort este de preferat sa folositi baze puteri a lui 2 și operatii pe biti. De asemenea daca nu folositi operatii pe biti este bine sa puteți face baza o constanta ușor de modificat și sa aveti 2 versiuni ale radixSort cu 2 baze diferite și să arătați cum influențează timpul de rulare modificarea bazei.