**--=-- STRUCTURI DE DATE an 1 CB -=-**

-=- **LABORATOR 10 SD** -=-

**– Heap –**

**Fișiere utilizate:**

* **heap.h – definiţia heap-ului şi definiţiile funcţiilor de alocare, distrugere şi afisare heap.**

*typedef int (\*TRel)(int, int);* /\* relatie intre 2 informatii \*/

typedef struct Heap

{

int nMax, ne; /\* numar maxim de elemente posibil,

respectiv numar efectiv de elemente din heap \*/

*int \*v;* /\* adresa de inceput a vectorului (alocat dinamic)

prin care este reprezentat heapul \*/

*TRel rel;* /\* functie de comparatie a doua elemente de tip int –

folosita pentru implementarea max/min heap \*/

*} \*THeap;*

* **heap.c – exemplificare heap (Observaţie – pentru construcţia heap-ului este folosit un vector de elemente întregi, ce respectă condiţia de heap).**

**Teme pregătitoare pentru laborator**

Să se implementeze următoarele functii:

**int Parinte(THeap h, int indCopil)**

/\* întoarce părintele copilului cu indicele indCopil \*/

**int FiuStanga(THeap h, int indParinte)**

/\* întoarce indicele fiului stâng al parintelui cu indice indParinte \*/

**int FiuDreapta(THeap h, int indParinte)**

/\* întoarce indicele fiului drept al parintelui cu indice indParinte \*/

Pentru nodurile inexistente se va folosi macrodefinitia NodInex.

**Observaţie**: indicii nodurilor heapului sunt numerotaţi de la 0

**Teme pentru laborator:** Se vor urmari TODO-urile din scheletul de cod.

Se puncteaza: 2p, 3p, 3p, 2p. Bonus 2p.

1. void PushHeap(THeap h, int val) – insereaza valoarea val in heap
2. int PopHeap(THeap h) – intoarce valoarea elementului eliminat
3. void MakeHeap(THeap h) – restabileste structura de heap fara a folosi memorie auxiliara
4. void HeapSort(int \*v, int nv, TRel rel) – sorteaza vectorul v dupa regula data de *TRel rel.*

**Bonus:** void PushHeap(THeap h, void \*info) Inserare în heap generic (definit pe baza unui vector generic). Se va folosi un vector cu elemente generice (se modifică funcţia de alocare şi eliberare heap).