

NUME:

PRENUME:

GRUPA:

	Ex1	Ex2	Ex3	Ex4
a				
b				
c				
d				

*Punctajele se acorda astfel: a=1p, b=2p, c=3p, d=4p, total  $4 \times (1+2+3+4) = 40p$*

*Punctajul pentru d se acorda doar daca ati rezolvat corect punctul c corespunzator*

## 1. Vectori si Hashtable

- Alocati dinamic si apoi eliberati memoria pentru un vector de tip si marime la alegere.
- Enumerati 3 avantaje si 3 dezavantaje ale unui hashtable fata de un vector obisnuit.
- Fie clasa Hashtable, ce modeleaza un hashtable cu rezolvarea conflictelor prin inlantuire directa. Implementati functia de adaugare si cea de interogare:

```
void put(Tkey key, Tvalue value)
Tvalue get(Tkey key)
```

- Propuneti o functie de hash pentru stringuri si estimati numarul de atribuii ce vor fi necesare pentru a adauga perechi (*nume\_student*, *nota\_student*) intr-un catalog al grupei voastre, implementat ca hashtable ce foloseste aceasta functie

## 2. Arbori Binari de Cautare

- Cum arata un arbore binar de cautare in care se introduce urmatoarea succesiune de chei: 20 10 22 15 8 16 23 18 19?
- Scriti functia de parcurgere in preordine unui arbore binar si ce va produce aceasta pentru arborele binar de cautare de la punctul precedent.
- Ce face functia de mai jos? Dati un exemplu pe arborele de la punctul a.

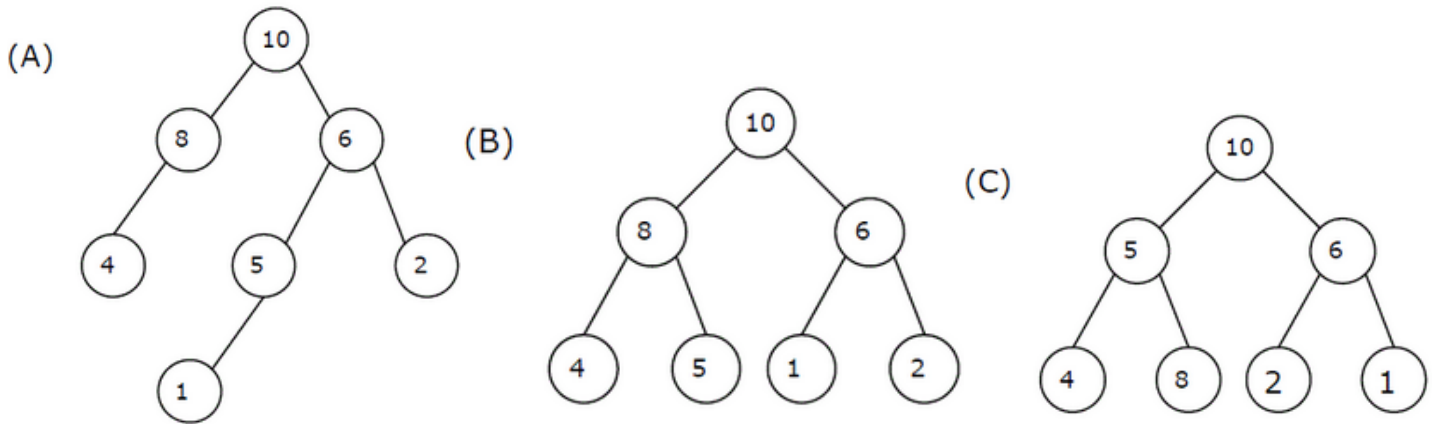
```
// A BST node
struct node {
    int data;
    struct node *left, *right;
};
int count = 0;
void print(struct node *root, int k)
{
    if (root != NULL && count <= k)
    {
        print(root->right, k);
        count++;
        if (count == k)
            printf("%d ", root->data);
        print(root->left, k);
    }
}
```

- Dati un exemplu de metoda de echilibrare a arborilor binari de cautare si explicati cum functioneaza asupra arborelui de la punctul a. Considerand 4 structuri de date - un arbore binar de cautare echilibrat, unul dezechilibrat, o lista dublu inlantuita sortata si una nesortata - comentati cu privire la efortul de procesare al urmatoarelor functii:

- functia de la punctul c
- functia de cautare a unei element

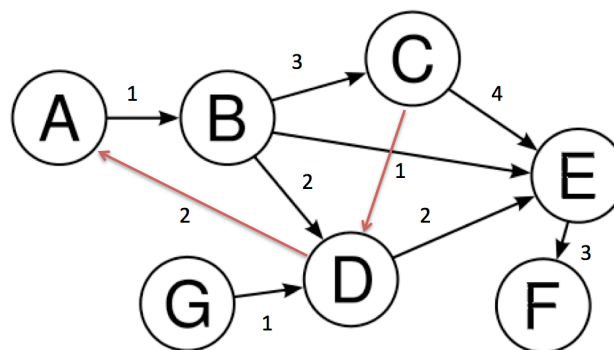
Comentati cu privire la necesitatea echilibrarii arborilor binari de cautare

### 3. Heap



- Care dintre arborii din figura sunt heap si care nu? Argumentati pentru fiecare.
- Fie un max-heap gol. Se adauga urmatoarele chei **13, 10, 7, 23, 9, 18, 8**. Apoi se elimina doua elemente. Desenati cum va arata heapul in cursul acestor operatii.
- Scrieti functia de adaugare in heap, inclusiv functia pushUp.
- Considerand o clasa de heap gata implementata (`template<typename T> class Heap`), implementati o clasa `PriorityQueue` care sa foloseasca un membru de tip `Heap` pentru a oferi functionalitatea unei cozi de prioritati si scrieti un exemplu de utilizare a clasei `PriorityQueue`.

### 4. Graf



- Reprezentati graful orientat din figura sub forma de lista de vecini.
- Care sunt componentele tare conexe ale grafului?
- Implementati o parcurgere DFS. Ce rezultat va produce pornind din nodul A, tinand cont ca vecinii se viziteaza in ordinea in care apar in lista de vecini de la punctul a.
- Explicati, prin exemple si cod, cum functioneaza unul dintre algoritmi de sortare topologica (nu uitati sa precizati cum se numeste algoritmul ales).