Laboratorul 5

Arbori. Arbori binari.

Matematic, un arbore este un **graf neorientat conex aciclic**.
În ştiinţa calculatoarelor, termenul de arbore este folosit pentru a desemna o structură de date care respectă definiţia de mai sus, însă are asociate un **nod rădăcină** şi o **orientare înspre sau opusă rădăcinii.**

Arborii sunt folosiţi în general pentru a modela o ierarhie de elemente. Astfel, fiecare element (nod) poate deţine un număr de unul sau mai mulţi descendenţi, iar în acest caz nodul este numit **părinte al nodurilor descendente.**

Fiecare nod poate avea un singur nod părinte. Un nod fără descendenţi este un nod terminal, sau **nod frunză**. În schimb, există un singur nod fără părinte, iar acesta este întotdeauna **rădăcina arborelui (root).**

Un arbore binar este un caz special de arbore, în care fiecare nod poate avea maxim doi descendenţi: **nodul** stâng şi **nodul drept.**

```
typedef int Item;

typedef struct Link
 {
   Item  elem;
   struct Link *l;
   struct Link *r;
 } TreeNode;
```

PARCURGEREA ARBORILOR

Preordine

- Se parcurge rădăcina
- Se parcurge subarborele stâng
- Se parcurge subarborele drept

Inordine

- Se parcurge subarborele stâng
- Se parcurge rădăcina
- Se parcurge subarborele drept

Postordine

- Se parcurge subarborele stâng
- Se parcurge subarborele drept
- Se parcurge rădăcina

CERINŢE

1. Să se scrie definiţia completă a următoarelor funcţi de lucru cu arborii în fişierul **Tree.h**:

   *void Init(TreeNode **t, Item x)* – iniţializează un nod cu valoarea x

   *void Insert(TreeNode **t, Item x)* – adaugă un nod cu valoarea x în arbore. La sfîrşit, pentru fiecare nod, subarborele stâng va conține valori mai mici decât cea a nodului, iar cel drept va conține valori mai mari decât cea a nodului (principiul Arborilor Binari de Căutare).

   *void PrintPostorder(TreeNode *t)* – afişează arborele în postordine

   *void PrintPreorder(TreeNode *t)* – afişează arborele în preordine

   *void PrintInorder(TreeNode *t)* – afişează arborele în inordine

   *void Free(TreeNode **t)* – eliberează spaţiul de memorie alocat arborelui

   int Size(TreeNode t)* – returnează dimensiunea arborelui (numărul de noduri)

   *int maxDepth(TreeNode *t)* – returnează adâncimea maximă a arborelui

   (8 X 0.75 puncte = 6 puncte)

2. În fişierul **testTree.c,** să se scrie definiţia completă a funcţiei:
   *void mirror(TreeNode **t)*
   care construieşte oglinditul unui arbore binar.

(1.5 puncte)

3. În fişierul **testTree.c**, să se scrie definiţia completă a funcţiei:
*int sameTree(TreeNode *t1, TreeNode *t2)*
care verifică dacă doi arbori sunt identici (sunt formaţi din noduri cu valori identice, aranjate identic din punct de vedere al dispunerii în arbore).

(1.5 puncte)