Laboratorul 1

Ex 1. Înmulțirea “a la russe”. Se considera doua numere naturale nenule a si b. Pentru a obține produsul a × b se procedeaza astfel. Se scriu a și b unul langa altul, form ˘and sub fiecare cate o coloana, conform următoarei reguli: se împarte numărul de sub a la 2, ignorând fracțiile apoi se înmulțeste cu 2 numărul de sub b. Se aplică regula, până când numărul de sub a este 1. În final, se aduna toate numerele din coloana lui b care corespund,pe linie, unor numere impare în coloana lui a. Este acest algoritm finit? Determină întotdeauna produsul a × b? Câte ınmulțiri cu 2 sunt necesare?

Ex2. Se consideră un set de n monede identice, cu excepția uneia care are greutatea mai mică decât celelalte. Folosind o balanță simplă (pot fi comprate monedele între ele pentru a afla care care taler este mai ușor de nu putem ști greutatea exactă) să se identifice moneda cu greutatea mai mică folosind cât mai puține comparații

Ex3. Să se calculeze cifra de control a unui număr. Cifra de control se obține calculând suma cifrelor numărului și apoi suma cifrelor numărului obținut etc. până când se ajunge la o singură cifră.

Exemplu: 4348 -> 19 -> 10 -> 1

Ex4. Să se verifice dacă un număr natural este palindrom. Un număr natural este palindrom dacă citit de la dreapta la stânga este egal cu numărul citit de la stânga la dreapta.

Ex5. Să se verifice dacă un număr natural este perfect. Un număr natural este perfect dacă el este egal cu suma divizorilor săi strict mai mici decât el.

Exemplu: 28 este perfect deoarece divizorii săi însumați dau numărul 1+2+4+7+14=28

Ex.6 Să se determine toate numerele perfecte mai mici decât un n dat, folosind programul anterior pentru număr perfect.

Ex7. Să se afișeze primele *n* perechi de numere prime gemene. Două numere naturale impare consecutive și prime se numesc prime gemene.

Laboratorul 2

Ex1. Se dă un tablou de numere întregi. Scrieți o funcție care determină diferența maximă (în modul) dintre oricare două elemente ale vectorului.

Exemplu: tabloul (10, 15, -2, 13, 20, -10, -5) are diferența maximă 30.

Ex2. Se dă un tablou de numere întregi. Să se determine poziția de început și lungimea celei mai lungi secvențe de numere egale. Dacă există mai multe secvențe de lungime egale, se va determina secvența cu poziția de început minimă.

Exemplu: (1,2,2,2,2,2,3,4,7,2,3,1,1,1,1,1,4) se va determina poziția 1 și lungimea 5.

Ex3. Se dă un tablou de numere întregi. Să se rearanjeze elementele astfel încât cele pare să apară înaintea celor impare. În cadrul subsecvenței de numere pare, respectiv impare, elementele trebuie să apară în ordinea în care erau în tabloul inițial.

Exemplu: (2,3,4,6,8,0,7,2,1) va fi rearanjat în (2,4,6,8,0,2,3,7,1).

Ex4. Se dă un tablou de numere întregi. Să se verifice dacă există un element majoritar (cu numărul de apariții mai mare decat n/2).

Laboratorul 3

Ex1.
Descrieți conținutul fiecărei variabile la fiecare pas când **x** are valoarea **3**.

```
x <- n
        for k <- 0 to n-1 do
                j <- 0
                while j <= k do
                        x <- x + x
                        j <- j + 1
```

Pas	Instructiune	Memorie
1	X = n	X = 3
2	K = 0	X = 3, K = 0

Ex2. Precizați și demonstrați complexitatea următorilor algoritmi
2.a)

```
sum <- 0
        for i <- 0 to n-1 do
                for j <- i to n-1 do
                        for k <- 0 to 4 do
                        sum <- sum + 2
```

2.b) -- **TEMĂ**

```
sum <- 0
        for i <- 0 to n-1 do
                for j <- i to n*n-1 do
                        sum <- sum + 1
```

Ex3. Să se determine ordinul de creștere al valorii variabilei x (in functie de n) după executarea algoritmilor.
3. a)

```
x <- 0
for i <- 1 to n do
        for j <- 1 to i do
                for k <- 1 to j do
                        x <- x + 1
```

3.b)

```
x <- 0
i <- n
while i >= 1 do
        x <- x + 1
        i <- i / 2
```

3.c)

```
x <- 0
i <- n
while i >= 1 do
        for j <- 1 to n do
                x <- x + 1
        i <- i / 2
```

3.d)

```
x <- 0
i <- n
while i >= 1 do
        for j <- 1 to i do
                x <- x + 1
        i <- i / 2
```

3.e)

```
x <- 0
i <- 1
while i < n do
        x <- x + 1
        i <- 2 * i
```

Ex4. Să se determine termenul dominant şi ordinul de creştere pentru expresiile:

4. a) $2 \log n + 4n + 3n * \log n$;

4. b) $2 + 4 + 6 + \cdots + 2n$

4. c) $((n+1)*(n+3)) / (n+2)$

4. d) $2 + 4 + 8 + \cdots + 2^n$

Ex5. Să se arate că: $n! \in O(n^n)$

Laboratorul 4

Ex 1. Să se rezolve următoarele recurențe folosind Teorema Master:
1.a) T(n) = 3 * T(n / 2) + n ^2
1.b) T(n) = 2 * T(n / 2) + n
1.c) T(n) = 8 * T(n / 2) + n ^ 2

Ex2. Se consideră o scară cu n trepte. Să se stabilească numărul de moduri în care poate fi urcată scara efectuând pași de una sau două trepte.

Ex3. Scrieți câte o funcție recursivă pentru calculul funcțiilor propuse, precizând și demonstrând clasa de complexitate.
3.a)
x^n, unde x aparține numerelor reale și n >=0.
3.b)
Fn, termenul de indice n din șirul lui Fibonacci, unde n>=0.

Ex4. Scrieți o funcție recursivă care calculează produsul a doi întregi m și n (m,n >=0) folosind doar operații de adunare și scădere.

Ex5. Scrieți o funcție recursivă care calculează [log2 (n)], n cu natural folosind doar operații de adunare și împărțite întreagă.

Laboratorul 5

Ex 1. Rezolvati urmatoarele recurente folosind arbori de recursie:

1.a)

$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

1.b)

$T(n) = 3T(\lfloor n/2 \rfloor) + n;$

Ex 2. Se considera urmatorul algoritm pentru generarea permutarilor de ordin n,($n \geq 1$).

```
procedure permutari(k, x[0..n],n)
begin
        if k == 1 then
                for i <- 1 to n do
                        print x[i]
        else
                for i <- 1 to k do
                        swap(x[i], x[k])
                        permutari(k-1, x, n)
                        swap(x[i], x[k])
end
```

Ex3. Se consideră o scară cu n trepte. Să se stabilească numărul de moduri în care poate fi urcată scara efectuând paşi de una sau două trepte.

Laboratorul 6

Structuri de date
I **Reprezentare cu tablouri**

```
struct LLin {
        Elt tab[0..Max-1]
        int ultim
}
```

II **Reprezentare cu structuri inlantuite**

```
struct nod {
        Elt inf
        nod * succ
}
struct LLin {
        nod * prim
        nod * ultim
}
```

Ex 1. Sa se afiseze elementele unei liste simplu inlantuite in ordine inversa, fara a se utiliza spatiu de memorie suplimentar.

Ex 2.
Scrieti o functie care testeaza daca doua liste liniare simplu inlantuite sunt egale.
Exemplu: Listele $2 \rightarrow 3 \rightarrow 7 \rightarrow \varnothing$ si $2 \rightarrow 3 \rightarrow \varnothing$ sunt diferite. Listele $1 \rightarrow \varnothing$ si $1 \rightarrow \varnothing$ sunt egale.

Ex. 3. Se dau L1 si L2, doua liste liniare simplu inlantuite de numere intregi, ordonate crescator. Sa se scrie o functie care returneaza lista formata din elementele lui L1 si ale lui L2 ordonate crescator. Eventualele aparitii multiple ale unui element vor fi pastrate in lista rezultat.
Exemple:
Pentru listele L1 = $2 \rightarrow 2 \rightarrow 7 \rightarrow \varnothing$ si L2 = $1 \rightarrow 2 \rightarrow 8 \rightarrow \varnothing$, se va returna lista cu elementele in ordine crescatoare $1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 7 \rightarrow 8 \rightarrow \varnothing$.
Pentru listele L1 = $3 \rightarrow \varnothing$ si L2 = $\varnothing$, lista returnata va fi $3 \rightarrow \varnothing$.

Structuri de date
Stiva (LIFO)
- top(S)
- push(S,e)
- pop(S)
- esteVida(S)
- stivaVida()

Coada (FIFO)
- citeste(C)

- insereaza(C, e)
- elimina(C)
- esteVida(C)
- coadaVida(C)

Ex 4.
Fie doua stive s1 si s2. Implementati operatiile care simuleaza comportamentul unei cozi folosind doar cele doua stive S1 si S2.

Laboratorul 7

Ex 1. Sa se afiseze elementele unei liste simplu inlantuite in ordine inversa, fara a se utiliza spatiu de memorie suplimentar.

Ex 2. Scrieti cate un subprogram pentru fiecare din cazurile urmatoare:
a) obtinerea versiunii in oglinda a arborelui;
b) afisarea nodurilor de pe un nivel dat din arbore;
c) numararea nodurilor din arbore;
d) numararea nodurilor din arbore cu un singur descendent;
e) afisarea celui de al k-lea element din traversarea inordine a arborelui.

Ex 3. Scrieti cate un subprogram pentru fiecare din cazurile urmatoare:
a) preordine (RSD)
b) inordine (SRD)
c) postordine (SDR)

Ex 4. Un arbore binar este considerat balansat daca diferenta dintre nodurile sale de inaltime este cel mult 1.

Ex 5. Ce returneaza urmatoarea functie?

```
function mystery(x)
{
    if (x == NULL) then
        return 0;
    else
        return max(mystery(x->stg), mystery(x->drp));
}
```

Laboratorul 9

Definiti un tip de date pentru reprezentarea unui digraf. Scrieti subprograme pentru urmatoarele operatii cu digrafuri:

Ex1. Citind datele de la tastura creati un digraf.

Ex2. Calculul gradului interior al varfurilor.

Ex3. Calculul gradului exterior ar varfurilor.

Ex4. Verificati daca un digraf contine un varf “groapa”. Un varf este numit groapa daca orice j != i exista un arc (j,i) si nu exista (i,j). Daca exista un astfel de varf atunci va returna varful gasit. Daca nu, va returna -1.

Ex5. Transpuneti digraful obtinut la Ex1.

Ex6. Parcurgeti graful recursiv (matrici + liste).

Ex7. Afisati componentele conexe ale grafului D..

Laboratorul 10

Ex1. Dati un exemplu de secventa de intrare pentru care algoritmul de sortare prin interclasare are complexitatea timp $\Theta(n \log n)$, iar algoritmul de sortare prin insertie are complexitatea $\Theta(n)$.

Ex2. Scrieti un program care identifica cele mai mari k elemente dintr-un tablou.
Exemplu: pentru [2, 24, 14, 9, 33, 5, 51], k = 3, programul va afis, a 51, 33, 24.

Laboratorul 11

Ex1. Se da un arbore binar (memorat cu pointeri).

a) Scrieti o functie care verifica daca acest arbore este arbore binar de cautare.
b) Scrieti o functie care verifica daca acest arbore este arbore AVL.

Ex2. Scrieti o functie care determina valoarea minima dintr-un arbore binar de cautare.

Ex3. Determinati valorile cuprinse in intervalul [a, b] dintr-un arbore binar de cautare.

Ex4. Scrieti o functie care aduna la fiecare nod dintr-un arbore binar de cautare valorile mai mari decat valoarea curenta.

Ex5. Consideram un vector v cu n elemente, ordonat crescator. Scrieti o functie care creeaza un arbore binar de cautare echilibrat utilizand valorile din vector.

Ex6.Scrieti o functie care verifica daca exista o pereche de valori intr-un arbore de cautare echilibrat care sa aiba suma egala cu x. Complexitatea timp a functiei nu ar trebui sa depaseasca O(n) - numarul de valori din arbore. Se poate folosi O(log n) spatiu suplimentar.

Laboratorul 12

Ex1. Fie o tabela de dispersie cu m = 11 intrari si functia de dispersie h1(k) = k mod m.
Inserati urmatoarele chei {22, 1, 13, 11, 24, 33, 18, 42, 31}, in ordinea data in tabela de dispersie. Utilizati urmatoarele metode pentru rezolvarea coliziunilor:

Ex: inserare(hashTable, cheie, valoare)
h(22) = h1(22) = 22 mod 11 = 0
h(1) = 1 mod 11 = 1
a) Inlantuire h(k) = h1(k)

b) Adresare deschisa – examinare liniara:
h(k, i) = (h1(k) + i) mod m

c) Adresare deschisa – dispersie dubla:
h(k, i) = (h1(k) + ih2(k)) mod m, unde
h2(k) = k mod (m − 1) + 1

Ex2. Consideram o lista de persoane angajate la o companie. Fiecare angajat se afla in subordinea unei persoane (sef). Data o lista de persoane sub forma (angajat, sef), sa se identifice numarul de persoane din ierarhie asociate fiecarui sef.
Exemplu: pentru (Andrei, Mihai), (Ion, Mihai), (Mihai, Alexandru), (Alexandru, Alexandru) se va returna: Andrei - 0, Ion - 0, Mihai - 2, Alexandru - 3.

Ex3. Implementati functia de stergere dintr-o tabela de dispersie atunci cand coliziunile sunt rezolvate prin adresare deschisa.