EXAMEN STRUCTURI DE DATE

Subiectul 5

1 Se consideră expresia **T= A*B+C*D+E/F-G** dată în formă poloneză infixată (normală).
Se cere:

0,8p a) Să se transforme expresia **T** în formă poloneză inversă ilustrând pas cu pas stiva şi ieşirea;

0,2p b) Fie **FPI(T)** forma poloneză inversă obţinută la punctul a) şi val(A)=5, val(B)=1, val(C)=3, val(D)=2, val(E)=9, val(F)=3, val(G)=6. Să se arate evoluţia stivei în evaluarea expresiei FPI(T) şi să se determine val(T).

2 Fie **B(n)** mulţimea tuturor arborilor binari cu **n** noduri. Pentru **Arb** ∈ **B(n)**, notăm cu m_{Arb} numărul

2,0p de legături nule ale reprezentării arborelui **Arb**. Să se determine arborele $\mathbf{Arb}_{min}$ ∈ **B(n)** astfel încât m_{Arbmin} este minim posibil. Justificaţi răspunsul.

3 Lista simplu înlănţuită **LinkedList**:

0,2p • definiţie,

0,2p • exemplu,

1,0p • declaraţie pentru clasa C++ (folosind Templates)

0,6p • implementarea destructorului.

4 Căutarea folosind tabele de dispersii:

0,4p • exemplu

1,0p • declaraţie clasă C++(folosind templates),

0,6p • exemple funcţii de dispersie si clasificarea lor

5 Scrieţi o funcţie

1,0p

```
friend ostream& operator<<(ostream& o, Polinom<T>& p);
```

care afişează un polinom sub forma algebrică uzuală (ex. 7***X**^3 - 4***X**^1 + 2)

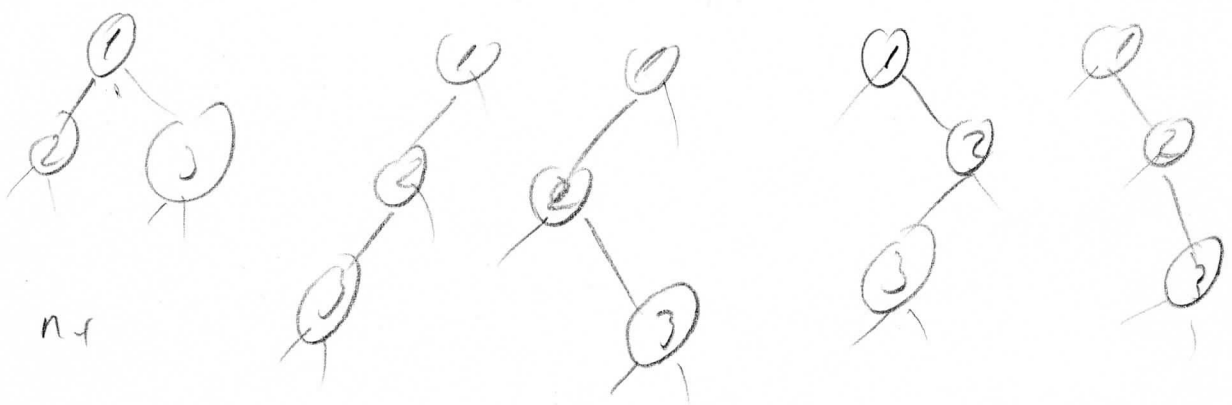
6 Pentru un arbore binar, scrieţi o metodă

1,0p

```
T BTree::getMax ()const;
```

care returnează valoarea maximă din arbore.

TOTAL: 9p+1p oficiu=10p



✓

① $T = A * B + C * D + E / F - G.$

$A = 5$

$B = 1$

$C = 3$

$D = 2$

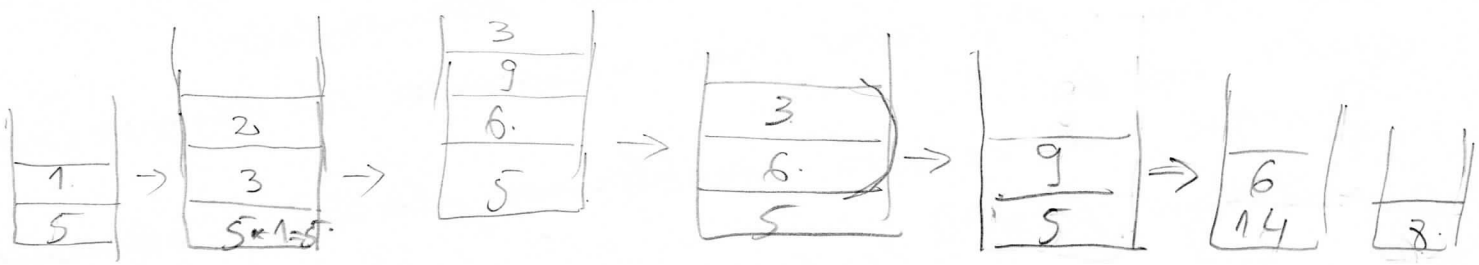
$E = 9$

$F = 3$

$G = 6.$

1) /
2) *
3) −
4) +

intrare	stiva	ieșire
A	VID	A
*	*	A
B	*	AB
+	+	AB*
C	+	AB*C
*	+ *	AB*C
D	+ (*)	AB*CD
+	+ +	AB*CD*
E	+ +	AB*CD*E
/	+ + /	AB*CD*E
F	+ + /	AB*CD*EF
−	−	AB*CD*EF/++
G	~~VID~~	AB*CD*EF/++G−

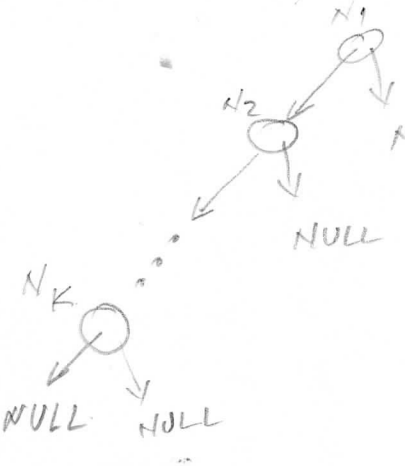


(2) Fie n nr. de noduri

Numărul legăturilor nule este $n(k-1)+1 \Rightarrow$

$\Rightarrow \boxed{n+1}$ legături nule. $\Rightarrow \underline{n-1}$ legături nenule.

$\Rightarrow$ arborele desfăcut (skewed)



$\Rightarrow$ arborele desfăcut :=

SKEWED

care funcționează ca o listă simplu înlănțuită

(4) Căutarea folosind tabele de dispersie

Fie M o mulțime de valori (chei) în nr. foarte mare.

Metoda de căutare folosind tabelele de dispersie constă în repartizarea valorilor din M în nr. $m \ll \text{card } M$ de containere folosind o funcție de repartiție. În acest fel vom restrânge aria de căutare numai la containerul corespunzător funcției de dispersie asociată valorii căutate.

Structura de date tabel de dispersie este un vector de containere, containerul de pe poziția k având memorate în el elementele mulțimii M care au valoarea de dispersie egală cu k.

Pentru a rezolva problema plasării elementelor mulțimii $M = \{x_1, x_2, \dots, x_m\}$ în tabela de dispersie vom folosi o funcție de dispersie.

$h : M \rightarrow \{1, 2, \dots dim_hash\}$

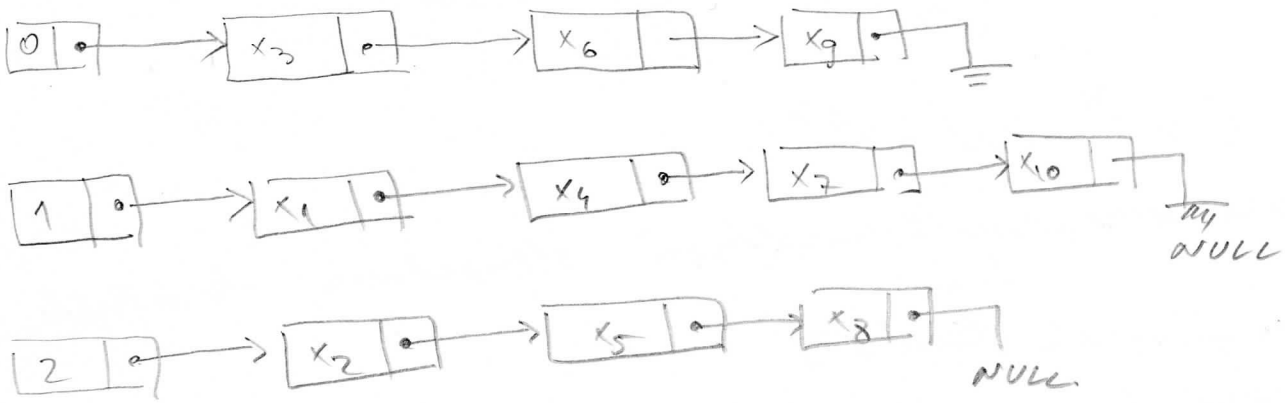
care ne furnizează pentru fiecare $x \in M$ capătul listei în care se află el.

Exemplu:

$M = \{x_1, x_2, \dots, x_{10}\}$, $h : M \to \{0, 1, 2\}$.

$h(x_i) = i \bmod 3$.

Avem



Obs. Funcția de dispersie este cu atât mai bună cu cât numărul de elemente din listă este relativ același. În acest caz spunem că avem o dispersie uniformă.

Există câteva clase de funcții de dispersie care asigură o repartiție cât mai uniformă.

(A) pentru dispersia valorilor întregi:

(i) metoda restului modulo.

(ii) metoda mediumul pătratului: care extrage numai biții mediani din ridicarea la pătrat a lui x.

Dacă $\dim \text{Hash} = 2^k$ și $W = 32$, atunci

$$h(x) = \lfloor 2^{k-W} (x^2 \bmod 2^{32}) \rfloor$$

(iii) metoda multiplicării: folosește aceeași idee ca cea anterioară pentru $a \cdot x$ în loc de x^2.

$$\Rightarrow h(x) = \lfloor 2^{k-W} (a x \bmod 2^{32}) \rfloor;$$

constanta a este aleasă astfel încât distribuția să fie uniformă.

(B) Pentru dispersia unui șir de caractere de lungime n, $s = s_0 s_1 \dots s_{n-1}$ se folosește funcția

$$h(s) = \left(\sum_{i=0}^{n-1} B^{n-i-1} \cdot s_i \right) \bmod W$$

$B = 2^8$, $W = 2^{32}$.

Clasa care implementează structura HashTable este

```
template < class T >
class HashTable : public SearchableContainer
{
protected:
unsigned int HASHSIZE;
Array <LinkedList<T>> hashTable
public.
HashTable (unsigned int hashSize : HASHSIZE (hashSize()){}
unsigned int isMember (T const &x) const.;
void insert (T &x);
void remove (T &x);
T & find (T const &x) const;
};
```

Construct

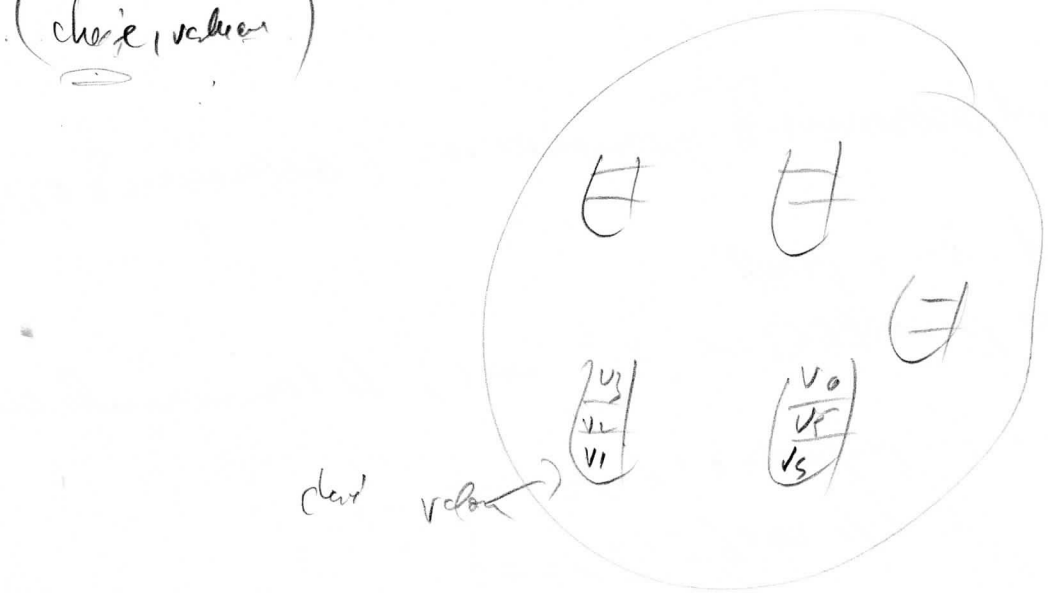
inserare

ștergere

căutare

5)

```
friend ostream & operator << (ostream & o, Polinom<T> p)
{
 int i = p.grad;
 o << p.coef[i] << "x^" << i; // termenul de grad maxim
 for (i = p.grad-1; i >= 1; i--)
     if (p.coef[i] >= 0)
         o << "+" << p.coef[i] << "*x^" << i;
     else
         o << p.coef[i] << "*x^" << i;
 if (p.coef[0] >= 0)
     o << "+" << p.coef[0] << endl;
 else
     o << p.coef[0] << endl;
 return o;
}
```


clef valeur