

1 Fie $B(n)$ un arbore binar cu n noduri. Spunem că $B(n)$ are proprietatea P dacă verifică una din următoarele două condiții:

- $n=1$;
- Dacă $n > 1$ atunci numărul de noduri din subarborele stâng al rădăcinii este cu exact 1 mai mare decât numărul de noduri din subarborele drept al rădăcinii și subarborii stâng și drept (dacă există) satisfac proprietatea P .

Se cere:

- Să se reprezinte $B(2)$ și $B(4)$;
- Să se demonstreze că pentru $n > 4$ nu există $B(n)$.

Handwritten notes:
 $n \geq 4 \Rightarrow 3$ nivele
 si nr impar de noduri din
 SS sau SD nu
 poate verifica propo
 iiii
 Cofect este o sa 2

2 2,0p Care este numărul de arbori binari cu exact n legături nenule? Demonstrație.

3 Permutarea cu implementare statică:

- exemplu
- structură de date C++ (cu Templates) pentru memorare (clasa **Permutare**)
- declarația și implementarea funcției de generare succesor.

4 Sortarea folosind algoritmul **Heapsort**:

- exemplu
- funcție C de inserare în heap
- calculul expresiei ordinului de operații

5 Se dă funcția $F: \mathbb{N} \rightarrow \mathbb{N}$ dată de relația

1,0p

$$F(x) = \begin{cases} 2F\left(\frac{x-1}{2}\right) + 1, & \text{dacă } x \text{ impar} \\ 2F\left(F\left(\frac{x}{2}\right)\right), & \text{dacă } x \text{ par, } x \neq 0 \\ 0 & \text{altfel} \end{cases}$$

Demonstrați că expresia lui $F(x)$ este corect definită, i.e. calculul lui $F(x)$ se termină într-un număr finit de pași oricare ar fi $x \in \mathbb{N}$.



6 Să se scrie o funcție

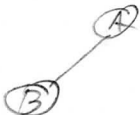
1,0p

int CircularQueueLength(CircularQueue* pq);

care determină lungimea unei cozi circulare memorate ca structură simplu înlănțuită.

TOTAL: 9p+1p oficiu=10p

Handwritten notes and diagrams:
 (2) nr. total de legături $2m$ de noduri
 $2mm - 1 = m + 1 \Rightarrow m = n + 1 \Rightarrow$
 \Rightarrow arbori binari care au $n+1$ noduri.
 (1) $B(2)$



① a).  B(2).

b)

② N_{r.} arborilor binari cu n noduri este

$$B(n) = \frac{1}{n+1} \cdot C_{2n}^n$$

Deoarece arborii au exact n legături
membru $\Rightarrow n-1 = n \Rightarrow n = n+1$.

Deci n_{r.} arborilor binari cu exact n legături este

$$\frac{1}{n+2} \cdot C_{2n+2}^{n+1}$$

⑤ Obs. că $F(k) = k \quad \forall k \in \mathbb{N}$.
inductie

③ Pomnitate cu implementare statică

```
class pomn
```

```
{
```

```
    int nr_elemente;
```

```
    int data[3];
```

```
public:
```

```
    pomn(); // constructor
```

```
    void create_c(int);
```

```
    pomn pomn_Next(pomn);
```

```
};
```

```
pomn::pomn() // constructor de initializare
{
    nr_elemente = 0;
}
```

```
void perm::create(int n)
```

```
{ nr_elemente = n;
```

```
for (i = 1; i <= nr_elemente; i++)
```

```
{ printf("data[%d] = ", i);
```

```
scanf("%d", &data[i]);
```

```
void perm::init() // initializare permutare  
identica.
```

```
{ for (int i = 1; i <= nr_elemente; i++)
```

```
data[i] = i;
```

```
}
```

```
perm perm: perm.Next(perm p)
```

```
{
```

```
int i = p.nr_elemente;
```

```
// caută un vârf.
```

```
while ((p.data[i] < p.data[i-1]) && (i > 1))
```

```
i--;
```

```
// a găsit pe i cu care trebuie să
```

```
aux = p.nr_elemente;
```

```
while (p.data[i-1] > p.data[aux]) // countă elem.  
aux--;
```

```
interschimba(p.data[i-1], p.data[aux]);  
// ordonează crescător. p[i], ... p[aux].
```

```
for (int piv = i; piv <= p.nr_elemente; piv++)
```

```
for (int j = piv + 1; j <= p.nr_elemente; j++)
```

```
if (p.data[piv] > p.data[j])
```

```
interschimba(p.data[piv], p.data[j])
```

```
nr_elemente = p.nr_elemente;
```

```
for (i = 1; i <= nr_elemente; i++)  
data[i] = p.data[i];
```