Nume _____ **Timp de lucru: 1h 40 min**

Grupa _____ Data _____

Analiza algoritmilor – Test 2

1. [3p] Orasul Chicago are multe cladiri, dar numai unele dintre ele au **vedere buna** catre lacul Michigan. Sa presupunem ca avem un vector A[1..n] care stocheaza inaltimile celor n cladiri din oras (acestea sunt indexate de la vest la est).

Cladirea cu numarul i are o vedere buna catre lacul Michigan **daca si numai daca** fiecare cladire de la estul ei este mai scunda. Se considera algoritmul prezentat in pseudocodul de mai jos care calculeaza care cladire are vedere buna catre lacul Michigan.

```
GoodView(A[1..n]) {
    S = stiva goala
    for (i = 1; i <= n; ++i) {
        insert(S, A, i); // insereaza in stiva elementul A[i]
    }
    // numerele ramase in stiva reprezinta indicii cladirilor cautate
    return S.getAllElements();
}
Insert(S, A, i) {
    while (S.empty() == false && A[i] > A[ S.top() ] {
        S.pop();
    }
    S.push(i);
}
```

Care este costul amortizat al operatiei Insert? Care este costul total al secventei de n operatii Insert? (al functiei GoodView).

Mentiuni:
- **top():** returneaza elementul din varful stivei fara a-l sterge din stiva;
- **pop()**: returneaza si elimina din stiva elementul din varf;
- **push(x)**: adauga elementul x in varful stivei
- **empty():** returneaza true daca stiva e goala; false daca stiva nu e goala

2.[1p] Fie contorul binar pe **k biti** studiat la seminar, care, pe langa operatia de incrementare, suporta acum si operatia de decrementare. Complexitatea unei secvente aleatoare de **n operatii** pe acest contor este:

a. O(n+k) **c. O(n*k)** **e. O(log n)**
b. O(n) **d. O(k)** **f. O(n log log k)**

3. [3p] Sa se precizeze efectul codului urmator si sa se demonstreze corectitudinea acestuia folosind un invariant la ciclare:

```
rev(v, n) { // input: vector v de dimensiune n
    i = (n-1)/2
    j = n/2
    while (i >= 0 and j <= n-1) {
        swap(v[i], v[j])
        i--
        j++
    }
}
```

4. [1p] Fie algoritmul **Bellman-Ford** de determinare a **drumurilor minime** de la o sursa **source** la orice nod **v** dintr-un graf cu muchii de costuri nenegative.

```
BellmanFord(vertices,  edges,  source) {
        for each v in vertices {
                d[v] = infinity
        }
        d[source] = 0
        n = size(vertices) // numarul de noduri din graf
        for i = 1 to n-1 {
                for each edge (u, v) with weight w in edges {
                        if (d[v] > d[u] + w) {
                                d[v] = d[u] + w
                        }
                }
        }
}
```

Sa se aleaga invariantul la ciclare corect pentru algoritmul de mai sus. Inainte de iteratia i:
a) d[v] este drumul minim de la source la v pentru v de la 1 la i-1
b) d[v] este al (n – i + 1) - lea cel mai scurt drum de la source la v
c) d[v] este drumul minim de la source la v care trece doar prin nodurile de la 1 la i-1
d) d[v] este drumul minim de la source la v care trece prin maxim i-1 muchii

Mentiuni:
- vertices = multimea de noduri din graf
- edges = multimea de muchii din graf
- source = sursa (nodul fata de care se determina drumurile minime)

5. [1p] Sa se defineasca constructorii de baza ai tipului pereche de numere naturale consecutive (in care primul numar e mai mic decat al doilea). Precizati numele, semnatura si o scurta descriere.

6. [3p] Se dau constructorii tipului arbore binar cu elemente de tip T (BinTree T).

Constructori:
nil : -> BinTree T
node : BinTree T * T * BinTree T -> BinTree T

Operatori:
nil? : BinTree T -> Bool
v : BinTree T -> Nat // numarul de noduri din arbore
e : BinTree T -> Nat // numarul de muchii din arbore

Axiome:
(N1) nil?(nil) = True
(N2) nil?(node(l,x,r)) = False

[0.5p] a) Sa se scrie axiomele operatorilor v si e.

[2.5p] b) Sa se demonstreze prin inductie structurala: **not(nil?(t)) => v(t) = e(t) + 1**

Observatie: La subiectele 2, 4 si 5 scrieti doar raspunsul. La celelalte subiecte este nevoie de o justificare completa a rezultatelor/afirmatiilor.