1. Rezolvati urmatoarele exercitii:

a) Algoritmul A este O(n), algoritmul B este $o(n^2)$, algoritmul C este ω(n),iar algoritmul D este $\Theta(n^2)$. Ce relatii de ordine putem stabili intre acesti algoritmi din punct de vedere al complexitatii? Scurta justificare.

b) Gasiti raspunsul urmatoarei egalitati (si justificare pentru acesta):

$\Omega(f(n)) + o(f(n)) = ...(f(n))$

2. Pentru fiecare afirmatie de mai jos, scrieti valoarea de adevar impreuna cu o scurta justificare sau demonstratia completa:

a) $f(n) = \theta(\mathrm{n})$ si $\mathrm{g(n)} = \Omega(\mathrm{n})$ atunci $\mathrm{f(n)} * \mathrm{g(n)} = \Omega(\mathrm{n}^2)$
b) $f(n) = \theta(1)$ atunci $\mathrm{n}^{\mathrm{f(n)}} = \mathrm{O(n)}$
c) $f(n) = \Omega(\mathrm{n})$ si $\mathrm{g(n)} = \mathrm{O}(\mathrm{n}^2)$ atunci $\frac{\mathrm{f(n)}}{\mathrm{g(n)}} = \mathrm{O(n)}$
d) $f(n) = O(n^2)$ *si* $g(n) = O(n)$ *atunci* $f\big(g(n)\big) = O(n^3)$
e) $f(n) = O(logn)$ atunci $2^{f(n)} = O(n)$
f) $f(n) = \Omega(\mathrm{n})$ atunci $2^{f(n)} = \Omega(n)$

3. Se da urmatoarea problema:

Fie n comutatoare ce corespund a n becuri (1..n), initial toate stinse. Se aplica urmatorii n pasi: la pasul k se modifica starea comutatoarelor din k in k incepand cu comutatorul 1 (1<=k<=n). Ne intereseaza cate becuri raman aprinse.

a) Scrieti pseudocodul algoritmului iterativ ce simuleaza cei n pasi descrisi anterior si determinati timpul sau de executie. Ce complexitate are acest algoritm (folositi notatia Theta) ?

b) Gasiti un algoritm cat mai eficient posibil pentru rezolvarea problemei. Indicatie: Ce pasi actioneaza asupra comutatorului k? Reduceti cat mai mult posibil complexitatea! Folositi de asemenea notatia Theta.

4. Se considera un tablou de n numere reale X[1..n]. Ne intereseaza sa raspundem la intrebari de genul: care este maximul subtabloului X[i..j]? Practic va trebui sa gasim un algoritm eficient per intrebare. Pentru aceasta vom imparti vectorul in n/k "bucati" de lungime k (eventual cu exceptia ultimei "bucati"). De exemplu pentru n = 8 si k = 3 vom avea [1..3], [4..6], [7, 8]. Pentru fiecare dintre aceste "bucati" vom calcula maximul si il vom retine intr-un vector max.

a) Scrieti in pseudocod un algoritm eficient ce gaseste maximul din subvectorul X[i..j]. Analizati complexitatea rezolvarii a m intrebari, luand in calcul si construirea vectorului max.

b) Cum ar trebui ales k astfel incat sa se obtina o complexitate cat mai mica ?

5. Intuiti o solutie si rezolvati prin metoda substitutiei recurenta: T(n) = T(n/2 + sqrt(n)) + n

6. Rezolvati urmatoarele recurente:
a) T(n) = T(sqrt(n)) + log n
b) T(n) = 2 * T(sqrt(n)) + 1

7. Fie tipul de date BTree definit prin constructorii:

BTempty: -> BTree
BTnode: BTree x T x BTree -> BTree

si definitiile:

flattenTree(t) : BTree -> List
(F1) flattenTree(BTempty) = []
(F2) flattenTree(BTnode(left, i, right)) = flattenTree(left) ++ [i] ++ flattenTree(right)

numBTelem(t): BTree-> N
(N1) numBTelem(BTempty) = 0
(N2) numBTelem(BTnode(left, i, right)) = 1 + numBTelem(left) + numBTelem(right)

length(l) : LIST -> N
(L1) length([]) = 0
(L2) length(h:t) = 1 + length(t)

Stiind ca avem constructorii [], [a] si h:t (echivalent cu cons(h,t)) pentru tipul LIST si ca operatorul ++ (concatenarea a doua liste) este definit astfel incat urmatoarea proprietate este adevarata pentru orice doua liste l1, l2: length(l1 ++ l2) = length(l1) + length(l2), sa se demonstreze ca:

numBTelem(t) = length (flattenTree(t)) pentru orice t de tip BTree

8. O expresie aritmetica complet parantezata este definita astfel:

0, 1, x, [e1+e2], [e1*e2], [-e2]

2 constructori nulari, 1 constructor extern, 3 constructori interni. Sa notam cu E acest tip de date.

Se definesc operatorii:

eval(e, n): E x N -> N
(E1) eval(0, n) = 0
(E2) eval(1, n) = 1
(E3) eval(x, n) = n
(E4) eval([e1+e2], n) = eval(e1, n) + eval(e2, n)
(E5) eval([e1*e2], n) = eval(e1, n) * eval(e2, n)
(E6) eval([-e1], n) = -eval(e1, n)

subst(e, f): E x E -> E
(S1) subst(0, f) = 0
(S2) subst(1, f) = 1
(S3) subst(x, f) = f
(S4) subst([e1+e2], f) = [subst(e1,f) + subst(e2, f)]
(S5) subst([e1*e2], f) = [subst(e1,f) * subst(e2, f)]
(S6) subst([-e], f) = [-subst(e, f)]

Sa se demonstreze prin inductie structurala ca pentru orice e, f din E si n din N, proprietatea urmatoare este adevarata:

eval(subst(e, f), n)= eval(e, eval(f, n))

9. Demonstrati corectitudinea sortarii prin selectie folosind invarianti la ciclare.

Notare: Fiecare exercitiu valoreaza 1p => 9 * 1p + 1p (oficiu) = 10p