1. (2p)

a. Scrieti un algoritm determinist care sorteaza eficient o lista. Justificati alegerea algoritmului folosit. Mentionati complexitatea acestuia.

b. Scrieti un algoritm nedeterminist a carui complexitate angelica este mai buna decat cea a algoritmului de la punctul a. Justificati expresia gasita a complexitatii, si comparatia intre cele doua complexitati.

c. Desenati arborele de evolutie a algoritmului nedeterminist, pentru un exemplu ales de voi, si evidentiati calea care se incheie cu succes.

d. Care este complexitatea algoritmului determinist asociat ? Justificati.

e. Care este complexitatea spatiala a algoritmului nedeterminist ? Justificati.

2. (1p) Clasa ArrayList din JAVA are urmatorul comportament (usor simplificat): Elementele din ArrayList sunt retinute intr-un vector intrinsec (cu capacitate initiala 2). Inserarea adauga elemente in acesta. Cand devine plin, vectorul este realocat, capacitatea acestuia sporind cu (parte intreaga din) jumatate din capacitatea anterioara. Elementele existente sunt copiate in noul vector.

   a. Ilustrati grafic evolutia vectorului intrinsec (si a capacitatii acestuia), pentru introducerea a 10 elemente intr-un obiect ArrayList.

   b. Ilustrati (completand desenul de mai sus) costul fiecarei operatii de inserare. Generalizati pentru o secventa de n inserari.

   c. Determinati costul amortizat al unei operatii de inserare. Justificati.

3. (2p) Dati cate un exemplu pentru fiecare din urmatoarele tipuri de probleme. Descrieti cate un algoritm pentru rezolvarea fiecarei probleme. Justificati:

   a. Problema nedecidabila
   b. Problema semidecidabila fara a fi decidabila
   c. Problema decidabila
   d. Problema NP-completa
   e. Problema intractabila, fara a fi NP-completa
   f. Problema tractabila

4. (2p) a. Construiti un algoritm ce realizeaza reducerea problemei k-ACOPERIRE la problema k-SET-COVER.

   k-ACOPERIRE: Se da un graf neorientat. Sa se determine daca exista o multime cu k noduri astfel incat toate muchiile din graf au cel putin un capat in aceasta multime.

   k-SET-COVER: Se da o multime (univers) U si mai multe submultimi Si incluse in U. Sa se determine daca exista k submultimi Si astfel incat reuniunea lor sa produca U. Determinati complexitatea acestuia.

b. Scrieti un algoritm nedeterminist care rezolva problema k-SET-COVER.
c. Presupunem ca problema k-ACOPERIRE este NP-completa. Demonstrati ca problema k-SET-COVER este NP-completa.
d. Presupunem ca problema k-SET-COVER este NP-completa. Ce putem spune despre problema k-ACOPERIRE ? Justificati.

5. (2p) Fie un tip de date Tree care modeleaza un arbore binar de cautare, si urmatorii constructori:
   Null:-> Tree
   Node:Tree x Int x Tree -> Tree
   a. Mentionati tipul acestor constructori.
   b. Explicati ce intelegeti prin “arbore binar de cautare”.
   c. Definiti urmatorii operatori:
      insert:Int x Tree -> Tree
      find: Int x Tree -> Bool
      Operatorul de inserare va adauga elemente in arbore, astfel incat proprietatea de “arbore binar de cautare” este respectata.
      Operatorul de cautare se va folosi de proprietatea de “arbore binar de cautare” pentru gasirea elementului dorit.
   d. Demonstrati, folosind inductia structurala, urmatoarele proprietati:
      find (x, insert(x,t)) = True (t – Tree, x – Int)
      find (x, insert(y,t)) = find(x,t) (t – Tree, x,y – Int distincte)

6. (1p) Dati cate 2-3 exemple de functii ce apartin multimilor O(sin n), Ω(sin n), Θ(sin n).