1.(3p) (a) Scrieţi un algoritm care primeşte ca parametri două tablouri cu valori întregi $x[1..n]$ şi $y[1..n]$ şi returnează numărul elementelor comune din cele două tablouri. (b) Scrieţi un algoritm care primeşte ca parametru o matrice $A[1..m, 1..n]$ şi determină perechea de linii (i, j) care au cele mai multe elemente comune (Indicaţie: se poate folosi algoritmul de la pct. (a)). (c) Stabiliţi ordinul de complexitate al algoritmului descris la punctul (b);
2.(2p) (a) Se consideră o stivă `s` pentru care au fost deja implementate operaţiile `push`, `pop`, `empty`. Propuneţi un algoritm care foloseşte stiva pentru a afişa în ordine inversă un şir de caractere preluat caracter cu caracter (în afara stivei algoritmul poate folosi doar o variabilă de tip caracter). (b) Se consideră o listă simplu înlanţuita `L1` specificată prin adresa de început (`L1.prim`) şi se consideră că pentru un element referit prin `ref` următorul element este referit prin `ref.urm`, iar valoarea asociată este specificată prin `ref.val` şi este un număr. Descrieţi un algoritm care construieşte o nouă listă simplu înlănţuită `L2` care conţine doar elementele pozitive din `L1` (nu contează ordinea în care sunt elementele din `L2`). **3.**(3p) Se consideră o funcţie $f : [a, b] \to R$ care are proprietatea că e continuă, monotonă şi $f(a)f(b) < 0$ şi următorul algoritm recursiv:

alg(x,y)
  if $y - x < \epsilon$ **then return**$((x + y)/2)$
  else
    $m \leftarrow (x + y)/2$
    if $f(x) * f(m) \leq 0$ **then return alg**(x, m)
    else return alg(m, y) **endif**
  endif

(a) Ce returnează algoritmul când este apelat pentru a şi b (ϵ este o valoare pozitivă mică)? (b) Considerând operaţia de înmulţire ca fiind dominantă să se scrie relaţia de recurenţă care descrie evoluţia timpului de execuţie $T(n)$ unde $n = \lfloor (b - a)/\epsilon \rfloor$. (c) Estimaţi ordinul de complexitate a algoritmului.
4.(2.5p) (a) Scrieţi un algoritm afişează toate submulţimile cu 3 elemente ale unei mulţimi. (b) Scrieţi un algoritm care afişează toate submulţimile cu k elemente ale unei mulţimi.
5.(2.5p) Se consideră un set de n activităţi care utilizează aceeaşi resursă. O activitate A_i începe la momentul s_i şi se finalizează la momentul $t_i > s_i$ (activitatea A_i se desfăşoară în intervalul $[s_i, t_i)$). Două activităţi sunt considerate compatibile dacă intervalele lor de desfăşurare sunt disjuncte.(a) Scrieţi un algoritm, bazat pe tehnica greedy, care determină numărul maxim de activităţi compatibile. (b) Stabiliţi ordinul de complexitate al algoritmului propus.

1.(3p) Se consideră o matrice $A[1..m, 1..n]$ având elemente din mulţimea $\{1, 2, \ldots, k\}$. (a) Scrieţi un algoritm care determină toate valorile distincte din matricea A (Indicaţie: se poate utiliza un tabel de frecvenţe)(b) Scrieţi un algoritm care determină cea mai frecventă şi cea mai puţin frecventă valoare din matrice (în cazul în care mai multe valori au frecvenţa maximă/minimă se vor determina toate) (c) Stabiliţi ordinul de complexitate al algoritmului descris la pct. (b);
2.(2p) (a) Se consideră o listă simplu înlănţuită care conţine termenii unui polinom (fiecare nod al listei conţine valoarea coeficientului (`coef`), gradul termenului (`grad`) precum şi referinţa către următorul element (`urm`). Scrieţi un algoritm care primeşte adresa primului element al listei precum şi un număr real x şi returnează valoarea polinomului în x. (b) Se consideră o stivă `S` care iniţial este vidă. Stabiliţi care este conţinutul stivei după efectuarea următoarelor operaţii:
`S.push(2); S.push(1); S.pop(); S.push(3); S.push(1); S.pop();`
3.(3p) Se consideră următorul algoritm recursiv:

alg(k)
if $k = 1$ **then** $x[k] \leftarrow 0$; **print** $(x[1..n])$
  $x[k] \leftarrow 1$; **print** $(x[1..n])$
else $x[k] \leftarrow 0$; **alg**$(k - 1)$
  $x[k] \leftarrow 1$; **alg**$(k - 1)$
endif

(a) Ce returnează algoritmul când este apelat pentru o valoare naturală nenulă n (în ipoteza că $x[1..n]$ este o variabilă globală)? (b) Considerând ca operaţie dominantă atribuirea de valori elementelor din tabloul x să se scrie relaţia de recurenţă care descrie evoluţia timpului de execuţie $T(n)$ (numărul de atribuiri) (c) Să se rezolve relaţia de recurenţă şi să se determine ordinul de complexitate a algoritmului.
4.(2.5p) Fie $a[1..m]$ un şir ordonat crescător, $b[1..n]$ un şir ordonat descrescător iar v o valoare. (a) Scrieţi un algoritm de complexitate $\mathcal{O}(\max\{m, n\})$ (folosind tehnica interclasării) care verifică dacă există cel puţin o pereche (i, j) cu proprietatea că $a[i] + b[j] = v$ (obs: este admis să se utilizeze tablouri adiţionale). (b) Justificaţi că algoritmul propus are complexitate liniară în raport cu m şi n.
5.(2.5p) Se consideră două secvenţe de numere naturale distincte: $[a_1, a_2, \ldots, a_n]$ şi $[b_1, b_2, \ldots, b_n]$. Propuneţi o modalitate de rearanjare a elementelor din cele două secvenţe astfel încât produsul $\prod_{i=1}^{n} a_i^{b_i}$ să fie maxim (produsul este specificat pe baza indicilor de după rearanjare). De exemplu pentru $a = [3, 5, 2]$ si $b = [1, 2, 3]$ produsul maxim este 2250. b) Stabiliţi ordinul de complexitat al algoritmului propus.