NR 1

1. (**4p**) Se consideră un vector de n elemente, numere naturale, v[0], v[1], ... ,v[n-1] cu proprietatea că există *i* astfel încât 0 < i < (n-1) şi v[0]<v[1]<...<v[i]>v[i+1]>...v[n-1]. Să se scrie un program care citeşte din fişierul date.in valoarea n urmată de vectorul v cu proprietatea din text, şi scrie pe ecran valorile i şi v[i]. Complexitate: **O(log n)** – abstracţie facând de citirea vectorului.

Spre exemplu, pentru fişierul date.in

```
8
2 4 5 9 10 12 9 7
```

se va afişa pe ecran

```
5
12
```

2. Fie n, m, p, q patru numere naturale si un şir $v_1, v_2, \ldots, v_n$ de numere naturale. Valorile n, m, p, q şi elementele vectorul v vor fi citite in ordine din fişierul date.in

a) Variantă pentru (3p) Să se afişeze pe ecran un subşir x al lui v de lungime maximă, *l*, cu proprietatea că suma sau diferenţa a două valori consecutive este egală cu p, modulo m, i.e $|x_i + x_{i+1}| \% m = p$ sau $|x_i - x_{i+1}| \% m = p$ pentru orice $i \in \{1,\ldots,l-1\}$. $\mathbf{O(n^2)}$

b) Variantă pentru (5p) Să se afişeze pe ecran un subşir x al lui v, de lungime maximă, *l*, cu proprietatea că valorile sale modulo *m* formeaza un şir în care există exact *q* punte de minim. Un punct de minim corespunde unui indice $i \in \{2,\ldots,l-1\}$ a.î $x_{i-1} > x_i < x_{i+1}$. sau $i = 1$ a.î $x_1 < x_2$ sau $i = l$ a.î $x_{l-1} > x_l$. $\mathbf{O(q * n^2)}$

Spre exemplu, pentru fişierul date.in

```
8 10 5 2
6 2 7 13 18 8 12 3
```

se va afişa pe ecran

```
a) 2 13 8 3
b) 6 2 7 18 3
```

pentru subşirul 2 13 8 3 avem $|2 + 13| \% 10 = |13 - 8| \% 10 = |8 - 3| \% 10 = 5$

subşirul (6 2 7 18 3) mod 10 = (6 2 7 8 3), are punctele de minim **2** şi **3**

Observaţii.

1. Pentru a promova testul de laborator **este <u>obligatoriu ca una dintre probleme să fie corect rezolvată.</u>**
2. Se acordă **1p** din oficiu.
3. Se va preda un director cu numele **grupa_Nume_Prenume** (se pot pune in director proiectul sau doar fişiere sursă, se pot rezolva cerinţele a si b in metode/clase/fişiere separate)

Nr 1.

P1. (5p) Se consideră o tablă dreptunghiulară cu *nxm* (*n, m* date) careuri. Pe orice careu al tablei este marcat un număr de puncte (număr întreg). Pe tablă se va deplasa un robot care are un număr inițial de puncte și care va pierde sau va câștiga puncte, adunând numerele marcate în careurile vizitate. Din celula (i,j) robotul se poate deplasa în celula (i+1, j) sau în celula (i,j-1). Robotul va putea vizita un careu doar dacă a acumulat până în acel moment un număr strict pozitiv de puncte.

Se citesc din fișierul de intrare **date.in** n și m și sfinal precum și matricea reprezentând tabla dreptunghiulară de dimensiune *nxm*. Să se scrie în fișierul **date.out** numărul inițial minim de puncte ***smin*** cu care robotul trebuie să înceapă un traseu din celula (1,m) pentru a ajunge în celula (n,1) având la finalul traseului valoarea mai mare sau egală cu ***sfinal***. Se va afișa de asemenea și un traseu care pornește din (1,m) și ajunge în (n,1) cu valoarea inițială ***smin*** la începutul traseului. **O(nxm)**

Exemplu:

date.in	date.out
3 3 13 2 -3 -10 -30 6 -5 -5 15 4	14 -10 -3 6 15 -5

Pentru n = 3, m= 3, sfinal = 13 și matricea de mai sus se va afișa ***smin*** = **14** și **traseul -10 -3 6 15 -5.** La finalul traseului robotul va avea suma 17.

VARIANTĂ pentru 2p: Să se rezolve problema în ipoteza că pe tablă sunt marcate doar numere pozitive.

P2. (4p) Se consideră o tablă pătratică de dimensiuni 2^N*2^N pe care sunt scrise numere naturale cuprinse între *1* si 2^N*2^N prin vizitarea recursivă a celor patru cadrane ale tablei în ordinea: stânga-sus, dreapta-jos, dreapta-sus, stânga-jos. De exemplu, daca *N=2*, tabla este completată astfel:

1 3 9 11
4 2 12 10
13 15 5 7
16 14 8 6

Fiind dat un număr $1 <= nr <= 2^N*2^N$ să se afișeze care sunt coordonatele acestuia pe tablă. Spre exemplu pentru datele de intrare N=2 și nr=12 se va afișa (2,3): numărul 12 este scris pe linia 2 coloana 3. **O(N)**

date.in	date.out
2 12	2 3

Observații.

1. Pentru a promova testul de laborator **este <u>obligatoriu ca una dintre probleme să fie corect rezolvată.</u>**
2. Se acordă 1p din oficiu.
3. Dacă există mai multe soluții, este suficient să se afișeze una
4. Se va preda un director cu numele **grupa_Nume_Prenume** (se pot pune in director proiectul sau doar fișiere sursă, se pot rezolva cerințele a si b in metode/clase/fișiere separate)

1.(**3,5p**) Descrieți metoda de programare *Divide et Impera*. Propuneți un algoritm cu complexitatea **O(log(n))** bazat pe această metodă pentru rezolvarea următoarei probleme (descriere, pseudocod, justificarea corectitudinii și **complexității** algoritmului propus, ilustrarea pașilor algoritmului pentru exemplul dat în enunț): *Se dau doi vectori a și b, fiecare având n elemente ordonate crescător. Să se determine mediana vectorului obținut prin interclasarea celor doi vectori.*
De exemplu: pentru n = 4 și vectorii [1 3 4 6][2 5 17 18], mediana vectorului obținut prin interclasare va fi 4,5 (media aritmetica a celor doua elemente din mijlocul acestui vector).

2. **(3.5p) a)** Se dau două șiruri de caractere A și B formate doar din litere mici ale alfabetului englez (de lungimi n, respectiv m). Să se determine lungimea maximă a unui subșir comun, LCS(A,B), a celor două șiruri, |LCS(A,B)|, și să se afișeze un subșir comun de lungime maximă |LCS(A,B)|. Propuneți un algoritm **O(nm)** bazat pe metoda programării dinamice pentru rezolvarea acestei probleme (pseudocod, complexitate, justificarea relației de recurență obținută pornind de la principiul de optimalitate, evidențiind subproblemele, relațiile de recurență și ordinea de calcul).
Exemplu: pentru șirurile A = *xty* și B = *axbcy* se va afișa |LCS(A,B)| = 2 și subșirului LCS(A,B) = *xy*.
(0.5p) b) Să se determine în câte moduri se poate adăuga o literă în șirul A, obținându-se șirul A', astfel încât |LCS(A,B)| = |LCS(A',B)| - 1, i.e **lungimea maximă a unui șir comun să crească cu 1**. |LCS(A,B)| se presupune precalculată. Evidențiați subproblemele, justificați **relația de recurență.**
Pentru A = *xty* și B = *axbcy* se poate adăuga una din literele *b*, *c* între *x* si *t* sau între *t* și *y* (**4** posibilități), obținându-se A'=*x***b***ty*, A'=*x***c***ty*, A'=*xt***b***y* respectiv A'=*xt***c***y* și subșirurile comune *xby* sau *xcy*.

3. (**1,5p**) Algoritmi genetici, descrieți etapa de selecție.

1.(**3,5p**) Descrieţi metoda de programare *Greedy*. Propuneţi un algoritm bazat pe această metodă pentru rezolvarea problemei continue (fracţionare) a rucsacului (enunţarea problemei, descrierea algoritmului, pseudocod, justificarea corectitudinii şi complexităţii algoritmului propus), date fiind capacitatea rucsacului W, greutăţile celor n obiecte $w_1,\ldots, w_n$ şi valorile $c_1,\ldots,c_n$ ale fiecărui obiect **pe unitatea de greutate (nu pentru întreg obiectul)**; datele de intrare sunt numere reale pozitive. Ilustraţi paşii algoritmului pentru W=9, n=4, w_1=3, w_2=3, w_3=4, w_4=2 şi c_1=4, c_2=3, c_3=2, c_4=5. Daţi un exemplu de date de intrare pentru care un algoritm de tip Greedy similar celui propus nu furnizează soluţia optimă pentru problema discretă a rucsacului.

2.(**4p**) Propuneţi un algoritm cu complexitatea **O(log(n))** bazat pe metoda *Divide et Impera* pentru rezolvarea următoarei probleme: Se dă un vector **a** cu **n** elemente distincte. Să se determine numărul de inversiuni semnificative din **a**. O inversiune semnificativă în vectorul **a** este o **pereche de elemente (a_i,a_j) cu proprietatea că i < j şi ai > 2 * aj.** Spre exemplu, pentru vectorul 4, 8, 11, 3, 5 numărul inversiunilor semnificative este 3: (8, 3), (11, 3), (11,5).

Variantă pentru (2p) Să se determine numărul de inversiuni dintr-un vector. O inversiune în vectorul a este o **pereche de elemente (a_i,a_j) cu proprietatea că i < j şi ai > aj.** Spre exemplu, pentru vectorul 4, 8, 11, 3, 5 numărul inversiunilor este 5: (4,3), (8, 3), (8,5), (11, 3), (11,5).

3. (**1,5p**) Algoritmi genetici, descrieţi etapele de crossover şi mutaţie.

Se dau n activități. Pentru fiecare activitate este cunoscut timpul de start s_i timpul de final f_i (intervalul de desfășurare $[s_i, f_i]$), precum și profitul p_i obținut în urma desfășurării activității i. Numim activități compatibile o submulțime de activități alese astfel încât intervalele de desfășurare să fie disjuncte două câte două.

a) Se citesc din fișierul **date.in** un număr natural, reprezentând numărul de activități si pe câte un rând s_i, f_i. Să se planifice o submulțime de activități compatibile de cardinal maxim. Se vor afișa activitățile planificate. **(4p – O(n log n))**.

4 4 7 1 3 2 6 10 11	1 3 4 7 10 11

b) Se citesc din fișierul **date.in** un număr natural, reprezentând numărul de activități si pe câte un rând s_i, f_i, p_i. Să se planifice o submulțime de activități compatibile de profit maxim. Se vor afișa profitul total și activitățile planificate. **(4p/5p – O(n^2)/O(n log n))**.

4 10 11 5 1 3 1 2 6 8 4 7 2	13 2 6 10 11

Observații.

1. Punctul b) se punctează doar dacă a) este corect rezolvat (astfel, pentru a promova testul de laborator **este <u>obligatoriu ca punctul a) să fie corect rezolvat</u>**).
2. Se pot folosi structuri de date existente în Java
3. Dacă există mai multe soluții, este suficient să se afișeze una
4. Se va preda un director cu numele **grupa_Nume_Prenume** (se pot pune in director proiectul sau doar fișiere sursă, se pot rezolva cerințele a si b in metode/clase/fișiere separate)

1. **(2,5p)** a) Descrieți metoda de programare Greedy. Propuneți un algoritm bazat pe această metodă pentru rezolvarea următoarei probleme (descrierea algoritmului, pseudocod, justificarea corectitudinii și complexității algoritmului propus). Se dau *n* activități numerotate 1,..., n. Pentru fiecare activitate *i* (cu $i \in \{1,..., n\}$) este cunoscut timpul de start și timpul de final: s_i respectiv f_i (activitatea *i* are intervalul de desfășurare $[s_i, f_i]$). Două activități sunt compatibile dacă au intervale de desfășurare disjuncte. Să se determine o submulțime de cardinal maxim de activități compatibile două câte două. Ilustrați pașii algoritmului propus pentru activitățile: [2, 4], [1, 5], [3, 7], [5, 6], [7, 8].

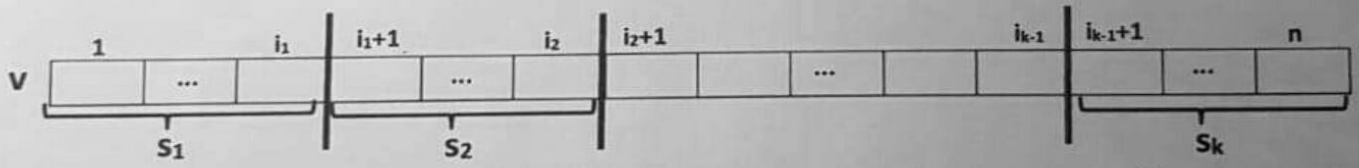
   b) Este corect următorul algoritm pentru rezolvarea acestei probleme? Justificați.

```
Fie A multimea activităților date
S ← ∅
Cat timp A ≠ ∅ executa
     Alege din A o activitate i care este compatibilă cu un număr maxim de
     activități din A
     Fie B mulțimea activităților din A incompatibile cu activitatea i
     S ← S ∪ {i}
     A ← A - {i}
     A ← A - B
Scrie S
```

2. **(2,5p)**Pentru un șir de *p* numere naturale $a=(a_1,\ldots, a_p)$ definim ponderea sa ca fiind $pond(a) = (a_p - a_1)^2$.
   Se dă un vector de ***n*** numere naturale $v=(v_1,...,v_n)$ și un număr natural **k ($1 \le k \le n$)** . O partiționare a lui v în k subsecvențe nevide (o **subsecvență** este formată cu elemente din vector aflate pe poziții consecutive) este de forma

$$s_1 = (v_1,\ldots, v_{i_1}),\ s_2 = (v_{i_1+1},\ldots, v_{i_2}),\ldots, s_{k-1} = (v_{i_{k-2}+1},\ldots, v_{i_{k-1}}),\ s_k = (v_{i_{k-1}+1},\ldots, v_n)$$

unde $1 \le i_1 < i_2 < \ldots < i_{k-1} < i_k = n$.



Ponderea unei astfel de partiționări este suma ponderilor subsecvențelor în care este partiționat vectorul: $pond(s_1) + \ldots + pond(s_k)$.
Să se determine o partiționare de pondere minima a lui v **în k** subsecvențe (nevide).
Propuneți un algoritm $O(\mathbf{n^2k})$ bazat pe metoda programării dinamice pentru rezolvarea acestei probleme (pseudocod, complexitate, justificarea relației de recurență obținută pornind de la principiul de optimalitate, evidențiind subproblemele, relațiile de recurență și ordinea de calcul)
Exemplu: pentru k=3, n=9 si v=(3, 1, 8, 7, 13, 12, 10, 5, 4) partiționarea în subsecvențele
$s_1 = (3, 1)$, $s_2 = (8, 7, 13, 12, 10, 5)$, $s_3 = (4)$ are ponderea
$pond(s_1) + pond(s_2) + pond(s_3) = (3 - 1)^2 + (5 - 8)^2 + (4 - 4)^2 = 4 + 9 = 13$.
O partiționare **de pondere minimă** în 3 subsecvențe a acestui vector este $s_1=(3, 1)$, $s_2=(8, 7, 13, 12, 10)$, $s_3=(5, 4)$ cu ponderea 4 + 4 + 1=9.

Precizare (variantă de 1p): O soluție corectă la această problemă bazată pe o altă tehnică se punctează **1p**

3. **(1p)** Algoritmi genetici – descrieți rolul etapei de selecție, detaliind selecția proporțională.

Observații
1. Se acordă un punct din oficiu
2. La punctajul obținut la examenul scris se adaugă punctajul obținut la temele de laborator