

MODALITATEA DE DESFĂȘURARE A EXAMENULUI LA DISCIPLINA "PROGRAMAREA ALGORITMILOR"

- Examenul la disciplina "Programarea algoritmilor" se va desfășura în ziua de 20.01.2022, între orele 9⁰⁰ și 11³⁰, astfel:
 - 09⁰⁰ – 09³⁰: efectuarea prezenței studenților
 - 09³⁰ – 11³⁰: desfășurarea examenului
 - 11³⁰ – 12⁰⁰: verificarea faptului că sursele trimise de către studenți au fost salvate pe platforma MS Teams
- Testul se va desfășura pe platforma MS Teams, iar pe tot parcursul desfășurării sale, de la ora 09⁰⁰ la ora 12⁰⁰, studenții trebuie să fie conectați pe canalul dedicat cursului de "Programarea algoritmilor" corespunzător seriei lor.
- În momentul efectuării prezenței, fiecare student trebuie să aibă pornită camera video în MS Teams și să prezinte buletinul sau cartea de identitate. Dacă dorește să-și protejeze datele personale, studentul poate să acopere codul numeric personal și/sau adresa!
- În timpul desfășurării testului studenții pot să închidă camera video, dar trebuie să o deschidă dacă li se solicită acest lucru de către un cadru didactic!
- Toate subiectele se vor rezolva folosind limbajul Python.
- Subiectul 1 este obligatoriu, iar dintre subiectele 2, 3 și 4 se vor rezolva CEL MULT DOUĂ, la alegere.
- Citirea datelor de intrare se va realiza de la tastatură, iar rezultatele vor fi afișate pe ecran.
- Se garantează faptul că datele de intrare sunt corecte.
- Operațiile de sortare se vor efectua folosind funcții sau metode predefinite din limbajul Python.
- Pentru subiectul 1 nu contează complexitatea soluției propuse.
- Rezolvările subiectelor alese dintre subiectele 2, 3 și 4 trebuie să conțină:
 - o scurtă descriere a algoritmului și o argumentare a faptului că acesta se încadrează într-o anumită tehnică de programare;
 - în cazul problemelor rezolvate folosind metoda Greedy sau metoda programării dinamice se va argumenta corectitudinea criteriului de selecție sau a relațiilor de calcul;
 - în cazul subiectelor unde se precizează complexitatea maximă pe care trebuie să o aibă soluția, se va argumenta complexitatea soluției propuse și vor primi punctaj maxim doar soluțiile corecte care se încadrează în complexitatea cerută;
 - în cazul problemei rezolvate folosind metoda backtracking nu contează complexitatea soluției propuse, dar se va ține cont de eficiența condițiilor de continuare;
 - în fiecare program Python se va preciza, pe scurt, sub forma unor comentarii, semnificația variabilelor utilizate.
- Rezolvările corecte care nu respectă restricțiile indicate vor primi punctaje parțiale.
- Se acordă 1 punct din oficiu.
- Rezolvările tuturor subiectelor se vor scrie de mână, folosind pix/stilou cu culoarea pastei/cernelii albastră sau neagră. Pe fiecare pagina studentul își va scrie numele și grupa, iar paginile trebuie să fie numerotate.
- Înainte de expirarea timpului alocat examenului, toate paginile vor fi fotografiate/scanate clar, în ordinea corectă, și transformate într-un singur fișier PDF care va fi încărcat în Google Drive folosind un anumit formular.
- Numele fișierului PDF trebuie să respecte șablonul *grupa_nume_prenume.pdf*. De exemplu, un student cu numele Popescu Ion Mihai din grupa 131 trebuie să denumească fișierul care conține rezolvările tuturor subiectelor astfel: *131_Popescu_Ion_Mihai.pdf*.

Subiectul 1 – limbajul Python – 3 p.

a) Unei liste formată din numere de la 1 la 26 i se asociază în mod unic un cuvânt astfel: numărul 1 este înlocuit de litera 'a', numărul 2 este înlocuit de litera 'b', ..., numărul 26 este înlocuit de litera 'z'. De exemplu, pentru lista [5, 23, 1, 13, 5, 14], cuvântul asociat este 'examen'. Să se scrie o funcție **cuvinte** care primește un număr variabil de liste formate din numere de la 1 la 26 și returnează un dicționar care conține, pentru fiecare listă, cuvântul asociat acesteia. De exemplu, pentru apelul **cuvinte([1, 14, 1], [1, 18, 5], [13, 5, 18, 5])**, funcția returnează dicționarul {'ana': [1, 14, 1], 'are': [1, 18, 5], 'mere': [13, 5, 18, 5]}. **(1.5 p.)**

b) Înlocuiți punctele de suspensie din instrucțiunea `lung = [...]` cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista să conțină toate tuplurile de forma (cuvânt, lungime) pentru fiecare cuvânt care începe cu o vocală dintr-o propoziție dată **p**. Propoziția este formată doar din litere mici ale alfabetului englez, iar cuvintele care o formează sunt distincte și despărțite între ele prin câte un spațiu. De exemplu, pentru propoziția `p = 'un exemplu de propozitie'`, lista rezultată va fi `lung = [('un', 2), ('exemplu', 7)]`. **(0.5 p.)**

c) Considerăm următoarea funcție recursivă:

```
def f(lista, p, u):
    if u - p <= 1:
        return lista[0]
    k = (p + u) // 2
    if k % 2 == 1:
        return f(lista, p, k-1) + f(lista, k+1, u)
    else:
        return f(lista, p, k-2) + f(lista, k+1, u)
```

Determinați complexitatea funcției apelată pentru o listă **L** formată din **n** numere întregi astfel: **f(L, 0, n-1)**. **(1 p.)**

Subiectul 2 – metoda Greedy (3 p.)

Complexitatea maximă a soluției: $O(n \log_2 n)$

La ora de sport, profesorul vrea să execute exerciții de gimnastică cu grupe de câte 2 elevi, dar pentru a putea realiza acest lucru trebuie ca valoarea absolută a diferenței dintre înălțimile celor 2 elevi dintr-o grupă să fie strict mai mică decât un număr natural h . Scrieți un program Python care citește de la tastatură două numere naturale n și h , precum și numele și înălțimile a n elevi, după care afișează pe ecran, în forma indicată în exemplu, numărul maxim de grupe formate din câte 2 elevi care se pot realiza respectând condiția indicată anterior, precum și numele elevilor din grupele respective. Evident, un elev poate să facă parte din cel mult o grupă! Înălțimile tuturor elevilor și diferența h sunt exprimate în centimetri. Nu contează ordinea în care se vor afișa grupele de elevi și nici ordinea numelor elevilor dintr-o grupă.

Exemplu:

Date de intrare	Date de ieșire
8 10 Popescu Ion 172 Mihai Ana 162 Popescu Dana 190 Ionescu Ion 181 Georgescu Ioana 170 Dumitrescu George 188 Constantinescu Radu 165 Georgescu Anca 210	3 Popescu Ion, Georgescu Ioana Mihai Ana, Constantinescu Radu Ionescu Ion, Dumitrescu George

Explicații: Avem $n = 8$ și $h = 10$. Se pot forma maxim 3 grupe de câte 2 elevi cu proprietatea că valoarea absolută a diferenței dintre înălțimile lor este strict mai mică decât 10 centimetri. Soluția nu este unică, o altă soluție corectă obținându-se, de exemplu, înlocuind grupa *Ionescu Ion, Dumitrescu George* cu grupa *Ionescu Ion, Popescu Dana*.

Subiectul 3 – metoda Programării Dinamice (3 p.)

Complexitatea maximă a soluției: $O(n^2)$

Schiorel a urcat cu telecabina până în vârful stațiunii și își dorește să ajungă cât mai obosit la una din cabanele stațiunii ca să se poată hidrata cât mai intens. Stațiunea e reprezentată ca o matrice pătratică de dimensiune n în care în fiecare pătrat avem gradul de oboseală pe care îl va acumula Schiorel dacă trece prin acel câmp sau -1, însemnând că în acel câmp avem o cabană. Schiorel poate începe traseul de oriunde de pe linia de sus și se poate opri la orice cabană, voi trebuie să-l ajutați să ajungă cât mai obosit! Schiorel poate coborî drept sau în diagonală, adică din (i,j) în $\{(i+1,j-1), (i+1, j), (i+1, j+1)\}$ evident fără a părăsi stațiunea.

Scrieți un program Python care citește de la tastatură dimensiunea tablei n și pentru fiecare pătrătică de coordonate (i,j) (cu $i=1,...,n, j=1,...,n$) o valoare c_{ij} cu semnificația:

- dacă c_{ij} este număr natural, el reprezintă gradul de oboseală acumulat de Schiorel când trece prin acea zona a stațiunii.
- dacă c_{ij} este -1, atunci în acea zonă se află o cabană!

și afișează un traseu al lui Schiorel până la o cabană, astfel încât să ajungă cât mai obosit (odată ajuns la o cabană, Schiorel se oprește și nu mai continuă drumul).

Intrare de la tastatură	Ieșire pe ecran
4 5 2 6 11 -1 7 1 -1 4 10 3 5 1 6 -1 2	Gradul de oboseala maxim 23 1 3 2 2 3 2 4 3

Explicații: Părtia este o matrice de dimensiuni 4x4 în care elementele reprezintă oboseala acumulată trecând prin acel punct, respectiv -1 în locul în care avem cabană. Pe traseul (1,3), (2,2), (3,2), (4,3) acumulează oboseala 23 (!traseul trebuie să înceapă pe prima linie și să se termine cu o pătrătică de valoare -1).

Intrare de la tastatură	Ieșire pe ecran
4 5 2 6 31 -1 7 -1 -1 4 10 3 5 1 6 -1 2	Gradul de oboseala maxim 31 1 4 2 3

Odată ajuns la o cabana, Schiorel se oprește din schiat.

P.S. Lui Schiorel îi place oboseala.

Subiectul 4 – metoda Backtracking (3 p.)

a) O țeavă cu lungimea de p metri ($1 \leq p \leq 50$) trebuie să fie tăiată în cel puțin două bucăți ale căror lungimi să fie divizori ai lungimii sale. De exemplu, o țeavă cu lungimea de 4 metri poate fi tăiată în 4 bucăți de câte 1 metru, 2 bucăți de câte 2 metri sau 2 bucăți de câte 1 metru și 1 bucată de 2 metri, dar nu poate fi tăiată într-o bucată de 1 metru și o bucată de 3 metri (deoarece 3 nu este un divizor al lui 4). Scrieți un program Python care să citească de la tastatură numărul natural p și afișează toate modalitățile distincte în care poate fi tăiată corect o bară de lungime p metri, precum și numărul acestora. Două modalități de tăiere se consideră identice dacă sunt formate din aceleași bucăți de țeavă, dar în altă ordine. De exemplu, pentru o țeavă cu lungimea de 4 metri, modalitățile de tăiere $1+1+2$, $1+2+1$ și $2+1+1$ sunt considerate identice. **(2.5 p.)**

Exemplu:

Pentru $p = 6$ trebuie afișate următoarele 7 modalități de tăiere (nu neapărat în această ordine):

1+1+1+1+1+1

1+1+1+1+2

1+1+1+3

1+1+2+2

1+2+3

2+2+2

3+3

Nr. modalitati: 7

b) Precizați cum ar trebui adăugată o singură instrucțiune în program astfel încât să fie afișate doar modalitățile de tăiere în care au fost utilizate exact două tipuri distincte de bucăți de țeavă. Pentru exemplul anterior, aceste soluții sunt cele scrise cu roșu. **(0.5 p.)**