

EXAMEN LA DISCIPLINA "PROGRAMAREA ALGORITMILOR"
– SESIUNEA Ianuarie 2025 – 1

Subiectul I (3 p.)

a) Scrieți o funcție **consoane_comune** care primește ca parametru un număr variabil de cuvinte (șiruri de caractere) și returnează numărul maxim de cuvinte dintre cele primite ca parametru cu proprietatea că multimea consoanelor pe care le conțin este aceeași (au aceeași mulțime de consoane).

De exemplu, pentru apelul **consoane_comune("este", "stea", "are", "rea", "tasta")** funcția trebuie să furnizeze valoarea 3, corespunzătoare cuvintelor "este", "stea", "tasta" care au aceeași mulțime de consoane. (1.5 p.)

b) Știind că **prefix_5** este o listă de numere naturale, înlocuiți punctele de suspensie din instrucțiunea **prefix_5 = [...]** cu o secvență de inițializare (*list comprehension*) astfel încât, după executarea sa, lista **prefix_5** să conțină toate numerele din lista **lista_numere** care nu sunt multiplii de 5 și nici nu au prefixe care sunt multiplu de 5 (prefixele unui număr se obțin din acesta eliminând cifre de la final: de exemplu, prefixele lui 123 sunt 12 și 1). De exemplu, pentru **lista_numere = [800, 246, 153, 12]**, lista **prefix_5** va fi **[246, 12]** (0.5 p.)

c) Considerăm următoarea funcție recursivă:

```
def f(s):
    if len(s)<=1:
        return s
    else:
        m = (len(s)-1) // 2
        c = s[m]
        if c in "aeiou":
            s = s.replace(s[m], " ")
        return f(s[:m+1]) + s[m+1:]
    else:
        return s[:m+1] + f(s[m+1:])
```

Determinați complexitatea maximă (în cel mai defavorabil caz) a funcției apelată pentru un sir de caractere **s** de lungime **n** astfel: **f(s)**. (1 p.)

$O(n)$

Subiectul 2 – metoda Greedy (3 p.)

Complexitatea maximă a soluției: $O(n * \log(n))$

Într-un garaj sunt $n \geq 1$ mașini și $m \geq 1$ canistre cu benzină. Pentru fiecare mașină se cunoaște numărul minim de litri de benzină M_i de care are nevoie pentru a ajunge la destinație, iar pentru fiecare canistră se cunoaște cantitatea de benzină C_i pe care o conține. Știind că unei mașini i se poate repartiza cel mult o canistră cu benzină, scrieți un program Python care citește de la tastatură numerele naturale $n, M_1, \dots, M_n, m, C_1, \dots, C_m$, după care afișează numărul maxim de mașini care pot ajunge la destinație, precum și o modalitate în care trebuie distribuite canistrelor mașinilor, în formatul din exemplul de mai jos (soluția nu este unică).

Exemplu:

Date de intrare	Date de ieșire
8	5
20 15 10 82 30 58 70 15	M1 -> C3
6	M2 -> C4
10 5 40 17 90 25	M3 -> C1
	M4 -> niciuna
	M5 -> C5
	M6 -> niciuna
	M7 -> niciuna
	M8 -> C6

Subiectul 3 – metoda Programării Dinamice (3 p.)

Complexitatea maximă a soluției: $O(k^*L)$

Un meșteșugar are o bară de metal de lungime L . El dorește să taie și să vândă bucăți din această bară, având comenzi pentru bucăți de lungimi specifice l_1, l_2, \dots, l_k . Fiecare lungime l_i are un preț asociat p_i cu care o poate vinde. Meșteșugarul poate vinde orice cantitate din fiecare dintre aceste lungimi. Ajutați-l pe meșteșugar să decidă cum să taie bară pentru a obține un câștig maxim, scriind un program Python care citește de la tastatură numerele L și k date pe o linie separate prin spațiu, apoi cele k lungimi l_1, l_2, \dots, l_k numere naturale date pe o linie separate cu spațiu, apoi cele k prețuri p_1, p_2, \dots, p_k numere naturale date pe o linie separate cu spațiu, ca în exemplu. Programul va afișa câștigul maxim pe care îl poate obține meșteșugarul și lungimile bucătilor pe care trebuie să le taie și să le vândă pentru a obține câștigul total maxim (având în vedere prețurile corespunzătoare fiecărei lungimi). Lungimile vor fi afișate în ordine crescătoare. Observație: se garantează faptul că întotdeauna meșteșugarul poate să vândă bucăți de lungime 1 (are comenzi pentru ele)!

Exemplu:

Date de intrare	Date de ieșire
9 5	47
1 2 4 5 6	1 4 4
5 6 21 22 23	

Explicație: meșteșugarul vinde o bucată de lungime 1 (preț 5) și două bucați de lungime 4 (fiecare cu preț 21) și obține câștigul maxim 47.

Subiectul 4 – metoda Backtracking (3 p.)

Gigel crede că un cod PIN format din $n \geq 3$ cifre este *mega-sigur* dacă toate cifrele sunt nenule, nu are cifre alăturate egale și orice cifră apare de cel mult p ori ($1 \leq p \leq n$). Scrieți un program Python care să citească de la tastatură două numere naturale n și p care respectă condițiile precizate mai sus și afișează toate codurile PIN mega-sigure formate din n cifre, precum și numărul lor.

Exemplu:

Pentru $n = 3$ și $p = 2$ trebuie afișate 576 de numere, o parte dintre ele fiind următoarele:
121, 123, 124, 125, 126, 127, 128, 129, 131, ..., 139, ..., 212, ..., 298, 312, 313,
314, 315, 316, 317, 318, 319, 321, 323, 324, 325, 326, 327, 328, 329, 341, 342, 343,
345, 346, 347, 348, 349, 351, 352, 353, 354, 356, 357, 358, 359, 361, 362, 363, 364,
365, 367, 368, 369, 371, 372, 373, 374, 375, 376, 378, 379, 381, 382, 383, 384, 385,
386, 387, 389, 391, 392, 393, 394, 395, 396, 397, 398, 412, ..., 498, 712, ..., 798,
812, ..., 898, 912, 913, 914, 915, 916, 917, 918, 919, 921, 923, 924, 925, 926, 927,
928, 929, 931, 932, 934, 935, 936, 937, 938, 939, 941, 942, 943, 945, 946, 947, 948,
949, 951, 952, 953, 954, 956, 957, 958, 959, 961, 962, 963, 964, 965, 967, 968, 969,
971, 972, 973, 974, 975, 976, 978, 979, 981, 982, 983, 984, 985, 986, 987, 989

NOTĂ:

1. Toate subiectele se vor rezolva folosind limbajul Python.
2. Subiectul 1 este obligatoriu, iar dintre subiectele 2, 3 și 4 se vor rezolva CEL MULT DOUĂ, la alegere.
3. Citirea datelor de intrare se va realiza de la tastatură, iar rezultatele vor fi afișate pe ecran.
4. Se garantează faptul că datele de intrare sunt corecte.
5. Operațiile de sortare se vor efectua folosind funcții sau metode predefinite din limbajul Python.
6. 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 încadreză 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 încadreză în complexitatea cerută;
 - în fiecare program Python se va preciza, pe scurt, sub forma unor comentarii, semnificația variabilelor utilizate.
7. Pentru subiectul 1 nu contează complexitatea soluțiilor propuse.
8. Rezolvările corecte care nu respectă restricțiile indicate vor primi punctaje parțiale.
9. Se acordă 1 punct din oficiu.