Examen scris la Algoritmică și Programare – 14.01.2020
Rândul 1

Nume și prenume: _____ Grupa: _____

Timp de lucru: **1 oră și 30 minute**. Fiecare subiect valorează **0.75 puncte**.

1. Completați spațiile subliniate astfel încât funcția de mai jos să returneze o listă de numere palindrom.

```
def palindroame(___, ___):
        return [___ for ___ in range(_____) if str(___)_____==_____(x)]
```

2. Scrieți o funcție care primește ca parametru o listă de string-uri și le afișează **scrise cu litere mari**.

3. Rescrieți funcția de mai jos folosind un **for** în loc de **while**:

```
def my_map(numbers, fun):
    result = []
    while numbers != []:
        result.append(fun(numbers[0]))
        numbers = numbers[1:]
    return result
```

4. Rescrieți funcția de la **punctul 3** folosind **list comprehensions**:

5. Rescrieți funcția de la **punctul 3** folosind **recursivitate**. Nu adăugați parametri suplimentari.

6. Scrieți o funcție **my_map2** care extinde funcția de la **punctul 2** pentru 2 liste date ca parametru și două liste returnate. Puteți folosi orice abordare.

7. Scrieți 3 aserțiuni pentru a testa funcția de la **punctul 6**.

8. Implementați funcția de mai jos conform specificațiilor:

```
def divizori(nr):
    # returnează o listă cu divizorii lui nr care conțin cifra 1
```

9. Completați specificațiile funcției de mai jos:

```
def pp(lista):
    # Returnează _____
    from math import sqrt
    return list(filter(lambda x: int(sqrt(x))*int(sqrt(x))==x, lista))
```

10. Considerăm un program scris obiectual folosind o arhitectură stratificată. Încercuiți toate afirmațiile **adevărate**:
    a. Sortările se vor face pe layer-ul de presentation
    b. Service-urile vor primi ca parametru în constructor maxim un repository
    c. Nu vom testa interacțiunea cu utilizatorul folosind assert-uri
    d. Clasele entitate pot conține setteri
    e. Obiectele valoare nu au identitate

11. Specificați complexitatea ca timp și spațiu suplimentar pentru funcția de mai jos, folosind notațiile O, Theta, Omega:

```
def f(n, k):
    if n > k:
        return True
    return False
```

Timp: Spațiu:

12. Implementați o clasă **Matrice2x2** care suportă **înmulțire prin operatorul *** și **calculul determinantului**

13. Asociați afirmațiile / conceptele cu paradigma de rezolvare a problemelor. Alegeți tot ce se aplică:

Subprobleme suprapuse	
Recursivitate	Backtracking
Deseori nu duce la o soluție optimă	Programare dinamică
Deseori ineficient	Divide et Impera
Subprobleme	Greedy
Formulă de recurență	

14. Asociați algoritmii cu complexitățile. Alegeți tot ce se aplică:

Mergesort	$O(n^2)$
Generarea tuturor permutărilor	O(n!)
Numărarea combinărilor folosind programare dinamică	O(n log n)

15. Scrieți o funcție **eficientă** care primește ca parametru o listă și returnează o listă formată din aceleași elemente, dar fără duplicate. De exemplu, f([2, 4, 2, 4, 3]) returnează [2, 3, 4] (sau orice altă ordine).