Examen scris la Algoritmică și Programare – 14.01.2020
Rândul 2

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(___, ___):
        if ___ > ___:
                return []
        if str(___) == ___(x)[::-1]:
                return [___] + palindroame(_____, ___)
        return _______________(x + 1, ___)
```

2. Scrieți un apel al funcției de la punctul 1 care dă eroare și explicați de ce dă eroare.

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

```
def my_factorial(n, step):
    result = 1
    for i in range(1, n, step):
        result *= i
    return result
```

4. Rescrieți funcția de la **punctul 3** folosind **list comprehensions**. Presupunem existența unei funcții **prod** care returnează produsul elementelor dintr-o listă.

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

6. Scrieți o funcție care calculează factorialul unui număr dat, folosind **două apeluri ale funcției de la punctul 3**.

7. Scrieți 4 aserțiuni pentru a testa funcția de la **punctul 6**.

8. Implementați funcția de mai jos conform specificațiilor:

```
def tribonacci(n):
    # implementează eficient formula: f(n) = f(n-1) + f(n-2) + f(n-3), f(n) = n pentru n < 3
```

9. Completați specificațiile funcției de mai jos:

```
def tribos(lista):
    # Returnează _____________________________________________________________________________
    from math import sqrt
    return list(zip(range(len(lista)), [tribonacci(x) for x in lista]))
```

10. Considerăm un algoritm backtracking. Încercuiți toate afirmațiile **false**:
    a. Pentru că este un algoritm backtracking nu poate efectua sortări
    b. Trebuie să conțină minim o căutare binară
    c. Poate fi implementat doar recursiv
    d. Reprezintă o căutare exhaustivă a unui spațiu, deci este foarte probabil să fie ineficient
    e. Va presupune revenirea la un pas precedent pentru a schimba o decizie

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 [n] * k
    return [k] * n
```

Timp: Spațiu:

12. Implementați o clasă **ȘirNumere** ce suportă **adăugarea elementelor printr-o metodă** și afișarea pe ecran.

13. Explicați, pe scurt, ce paradigmă de rezolvare a problemelor s-ar potrivi pentru rezolvarea **eficientă** a următoarei probleme: determinarea **numărului** de moduri în care se poate plăti o sumă dată folosind niște bancnote date.

14. Scrieți o funcție care are complexitatea timp $O(n^{2020})$:

15. Scrieți o funcție care primește ca parametru un dicționar cu chei întregi și valori string-uri și returnează o listă de tupluri (valoare, cheie). De exemplu, pentru dicționarul {2020: "anul nou"} se va returna [("anul nou", 2020)].