MAC0122 Princípios de Desenvolvimento de Algoritmos

BACHARELADO EM ESTATÍSTICA, MATEMÁTICA E MATEMÁTICA APLICADA

Primeira Prova – 1 de setembro de 2016

Nome: ____

Assinatura: ____

Nº USP: ________________ Prof: ___________________

Instruções:

1. Não destaque as folhas deste caderno. A prova pode ser feita a lápis.
2. A prova consta de 2 questões, cada uma com 2 itens. Verifique antes de começar a prova se o seu caderno está completo.
3. As questões podem ser resolvidas em qualquer página. Ao escrever uma solução (ou parte dela) em página diferente do enunciado, escreva QUESTÃO X em letras ENORMES junto da solução.
4. **As soluções devem ser em Python**. Você pode usar apenas recursos de Python **vistos em aula**. Você pode definir funções auxiliares e usá-las à vontade. Cuidado com a legibilidade e, principalmente, com a **tabulação**.
5. As soluções não precisam verificar consistência de dados.
6. Não é permitido o uso de folhas avulsas para rascunho, a consulta a livros, apontamentos, colegas ou equipamentos eletrônicos. Desligue o seu celular e qualquer equipamento que possa perturbar o andamento da prova.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1(a)	2,5	
1(b)	2,5	
2(a)	2,5	
2(b)	2,5	
Total	10,0	

Questão 1 (vale 5,0 pontos)

Está questão é composta de dois itens.

Item (a) (vale 2,5 pontos)

Escreva uma função **recursiva** `some_par_impar_recursiva()` que respeite a especificação a seguir.

```
def some_par_impar_recursiva(v, n):
    '''(list, int) -> int, int

    Recebe uma lista v de números inteiros e um inteiro n e
    retorna dois valores, a soma dos números pares e a soma
    dos números ímpares em v[0:n].

    Exemplos:
    >>> some_par_impar_recursiva([2,-5,7,12,15,7],2)
    2, -5
    >>> some_par_impar_recursiva([2,-5,7,12,15,7],3)
    2, 2
    >>> some_par_impar_recursiva([2,-5,7,12,15,7],4)
    14, 2
    >>> some_par_impar_recursiva([2,-5,7,12,15,7],6)
    14, 24
    '''
```

Item (b) (vale 2,5 pontos)

Escreva um programa (= função `main()`) que leia um inteiro $n \geq 0$ e uma sequência de `n` números inteiros e imprima a soma dos números pares e a soma dos números ímpares da sequência.

O seu programa **deve utilizar obrigatoriamente** a função `some_par_impar_recursiva()` do item (a). Você pode utilizar a função do item (a) mesmo que não a tenha feito.

A seguir está um exemplo de execução do programa:

```
Digite o número de elementos da sequência: 6
Digite o 1o. número: 2
Digite o 2o. número: -5
Digite o 3o. número: 7
Digite o 4o. número: 12
Digite o 5o. número: 15
Digite o 6o. número: 7
soma par = 14, soma ímpar = 24
```

Questão 2 (vale 5,0 pontos)

Nesta questão você escreverá duas funções que são simplificações bem grandes de funções que você fez para o seu EP3.

Em relação a um texto, o **dicionário dos `t`-gramas** é o dicionário em que as **chaves** são os `t`-gramas no texto e o **valor** associado a cada chave é o número de ocorrências do `t`-grama no texto.

As funções a seguir manipularão um dicionário dos `t`-gramas implementado por uma lista contendo duas listas, a **lista das chaves** e a correspondente **lista dos valores**. Assim, o dicionário vazio é `[ [], [] ]`.

Por exemplo, o dicionário dos 2-gramas do texto `"Teste é teste."` é

```
[[' t', ' é', 'Te', 'e ', 'e.', 'es', 'st', 'te', 'é '], [1, 1, 1, 1, 1, 2, 2, 3, 1]]
```

Item (a) (vale 2,5 pontos)

Escreva uma função `insira_ordenado()` que respeite a especificação a seguir.

```
def insira_ordenado(tgram, dicio):
    '''(str, dicio) -> None

    Recebe um t-grama tgram e um dicionário dos t-gramas dicio.
    Se tgram é uma chave no dicionário, então a função incrementa
    o valor correspondente, em caso contrário, a chave tgram é
    inserida no dicionário com valor 1.

    Importante: a função deve manter as chaves do dicionário em
       em ordem "alfabética" crescente.

    Exemplos:
    >>> dicio = [['es', 'te'], [1, 1]]
    >>> tgram = 'st'
    >>> insira_ordenado(tgram,dicio) # insira 'st' no dicionário
    >>> print(dicio)
    [['es', 'st', 'te'], [1, 1, 1]]
    >>> tgram = 'te'
    >>> insira_ordenado(tgram,dicio) # insira 'te' no dicionário
    >>> print(dicio)
    [['es', 'st', 'te'], [1, 1, 2]]
    >>> tgram = 'es'
    >>> insira_ordenado(tgram,dicio) # insira 'es' no dicionário
    >>> print(dicio)
    [['es', 'st', 'te'], [2, 1, 2]]
    '''
```

Item (b) (vale 2,5 pontos)

Escreva uma função `monta_t_gramas()` que respeite a especificada a seguir.

A sua função **deve utilizar obrigatoriamente** a função `insira_ordenado()` do item (a). Você pode utilizar a função do item (a) mesmo que não a tenha feito.

```
def monta_t_gramas(t, texto):
    '''(int, str) -> dicionário

    Recebe um inteiro t e um string texto e cria e retorna o dicionário
    dos t-gramas do texto.

    Exemplos:

    >>> monta_t_gramas(1,"testes")
    [['e', 's', 't'], [2, 2, 2]]
    >>> monta_t_gramas(2,"testes")
    [['es', 'st', 'te'], [2, 1, 2]]
    >>> monta_t_gramas(2,"tesTes")
    [['Te', 'es', 'sT', 'te'], [1, 2, 1, 1]]
    >>> monta_t_gramas(2,"tEsTes")
    [['Es', 'Te', 'es', 'sT', 'tE'], [1, 1, 1, 1, 1]]
    >>> monta_t_gramas(2,"tes tes")
    [[' t', 'es', 's ', 'te'], [1, 2, 1, 2]]
    >>> monta_t_gramas(1,"banana")
    [['a', 'b', 'n'], [3, 1, 2]]
    >>> monta_t_gramas(2,"banana")
    [['an', 'ba', 'na'], [2, 1, 2]]
    >>> monta_t_gramas(3,"banana")
    [['ana', 'ban', 'nan'], [2, 1, 1]]
    >>>
    '''
```