Instituto Federal do Espírito Santo
Campus Serra
Coordenadoria de Informática
Curso Superior Bacharelado em Sistemas de Informação

Prova Final de Programação 2 - Semestre 2015-2

Nome:_____ Turma:_____ Data: ___/___/_____

Leia com atenção o enunciado a seguir e faça o que é pedido e da forma como é pedido.

- **O material para a prova estará contido na pasta *paraProvaF.zip*, ou no quadro.**
- **Para a entrega da prova, compacte os arquivos .py em um arquivo .zip e faça a entrega como uma atividade no moodle. Nomeie o arquivo zip com *<seu nome>.zip*.**
- **Toda questão estará correta somente se produzir os resultados esperados e presentes nas figuras, obedecendo as especificações e restrições fornecidas no enunciado.**
- **Os enunciados das questões 3 e 4 abordam os conceitos presentes na atividade de janeiro sobre espaços vetoriais contextualizados.**
- **Em todos os enunciados, vetores são representados por listas de componentes do vetor.**
- **Em todos os enunciados, matrizes são implementadas como listas de listas, com as últimas representando as linhas da matriz. Matrizes não contém informação textual.**
- **Em caso de dúvidas, solicite a orientação do professor.**

Questão 1 (15 pontos)

Construa um arquivo python chamado questao1.py contendo funções que obedeçam totalmente às seguintes especificações (ver figura 1 para exemplos):

a) Assinatura e funcionalidade: ***separaPal(<texto>)***, recebe um texto qualquer como entrada e retorna uma lista de itens do texto, incluindo palavras simples, palavras compostas (com hífen na sua composição, a exemplo do que foi feito em sala de aula) e separadores (exceto espaço). Uma palavra é qualquer sequencia de letras e dígitos e hífen (sinal de menos). Separador é qualquer caractere diferente de letra e dígito. **Obs: espaço é o único separador que não deve constar na lista de saída.** (10 pts)

b) Assinatura e funcionalidade: ***ngramaPal(<lista palavras>,<tam>)***, recebe como entradas uma lista de palavras e um inteiro definindo o tamanho do ngrama. A função deve retornar a lista de todos os ngramas de palavras de tamanho <tam>. (5 pt)

c) Assinatura e funcionalidade: ***frequenciaTxt(<texto>)***, recebe como entrada um texto qualquer e retorna como saída uma tabela de frequência de ocorrência de tokens do texto, exceto separadores. Ou seja, frequências de ocorrência de palavras simples, palavras compostas e datas. (5 pt)

Questão 2 (25 pontos)

Construa um arquivo python chamado *tadpoli.py* que implementa o conceito de um tipo abstrato de dados (tad) polinômio. O poliônmio a ser implementado é do tipo daquele que possui apenas 1 variável. Ver a figura 1 para exemplos. O arquivo conterá funções que seguem totalmente às especificações listadas abaixo(ver figura 1 para exemplos):

a) *cria(<string contendo polinômio>)*: cria e retorna um tad polinômio implementado a partir de dicionário apenas. O polinômio em si é fornecido no formato textual string. Ver figura no quadro para a descrição do formato. Faz parte da questão observar as partes componentes do polinômio e decidir qual componente será chave e qual componente será o conteúdo do dicionário. (4 pontos)

b) *somapoli(<tadpoliA>,<tadpoliB>)*: soma os polinômios armazenados nos parâmetros de entrada. Retorna o polinômio resultado da soma. A soma obedece ás regras da álgebra de polinômios vistas em cálculo 1. (4 pontos)

c) *multipoli(<tadpoliA>,<tadpoliB>)*: multiplica os polinômios fornecidos pelos parâmetros de entrada. Retorna o polinômio resultado da multiplicação. A multiplicação obedece ás regras da álgebra de polinômios vistas em cálculo 1. (4 pontos)

d) *derivada(<tadpoli>)*: retorna o tad polinômio correspondente à derivada do tad polinômio fornecido como parâmetro de entrada. O cálculo da derivada obedece às regras da derivada de polinomiais. (4 pontos)

e) *toString(<tadpoli>)*: retorna uma string contendo a representação textual do tad polinômio fornecido como parâmetro de entrada. O formato da representação textual está descrito na figura fornecida pelo professor, no quadro. A função estará correta se o leiaute produzido for o mesmo fornecido pela figura. (4 pontos)

f) Construa uma aplicação que use o módulo *tadpolinomio* (*tadpoli.py*) e realize o seguinte processamento: ler o arquivo texto *bdpoli.dat*, linha por linha (usar apenas *readline()*). Cada linha do arquivo contém o seguinte formato: (5 pontos)

```
<operação>: <polinômio 1>, <polinômio 2> ou apenas
<operação>: <polinômio 1>

Exemplo, conteúdo hipotético de bdpoli.dat:
    soma: 2x4 – 4x3 + 2x2 + x + 5, - 5x2 + x
    subt: 5x3 + x2 + 8, 3x4 + 5
    deri: 4x8 + 5x6 + 3x2 + 12
    mult: 12x3 + 7x2 – 6, 12x2 + 3x - 4
```

Para cada linha lida, efetuar a operação pedida e salvar o conteúdo no arquivo texto de saída *poliout.txt.* As operações que aparecem no arquivo são: ***soma*** (soma de polinômios), ***subt*** (subtração de polinômios), ***deri*** (derivada de um polinômio) e ***mult*** (multiplicação de polinômios).

A aplicação estará correta se for construída corretamente e produzir os resultados corretos de acordo com as funções definidas nos itens a) a e).

Questão 3 (30 pontos)

Construa um arquivo python chamado *questao3.py.* O arquivo deverá conter as seguintes funções (utilize os mesmos nomes de funções fornecidos no enunciado):

a) *normalizaEV(<nome pasta espaço vetorial>)*: a função calcula e retorna uma lista contendo as dimensões do espaço vetorial composto (normalizado) referente à coleção de vetores (arquivos textos vetores) armazenados na pasta cujo nome é fornecido como argumento de entrada). Apenas para lembrar, o espaço vetorial normalizado é o conjunto de todas as dimensões (componentes/características/eixos propriedades) nas quais os vetores do espaço possuem componentes. (15 pts)

Construa uma aplicação de testes utilizando o espaço vetorial (pasta) *vetapartamentos*, disponibilizado como material de prova.

b) *matConfusao(<nome pasta espaço vetorial>)*: a função calcula e retorna a matriz de confusão do espaço vetorial armazenado na pasta cujo nome é fornecido como argumento de entrada). Apenas para lembrar, a matriz de confusão é uma matriz do tipo propriedades x vetores (linhas contém propriedades/dimensões/características e colunas contém vetores do espaço vetorial). (15 pts)

Condição necessária: *esta função tem de ser construída fazendo chamada à função normalizaEV. Somente após a obtenção das dimensões normalizadas ter ocorrido é que o processamentos dos vetores arquivos deve acontecer.*

Construa uma aplicação de testes utilizando o espaço vetorial (pasta) *vetapartamentos* fornecido como material de prova. Utilize a matriz de confusão da figura 3 como gabarito para a sua aplicação de testes.

Utilize as figuras fornecidas para a obtenção das fórmulas necessárias para a implementação da função.

$$\cos(\alpha) = \frac{\sum_{i=1}^{n} U_i V_i}{\sqrt{\sum_{i=1}^{n} U_i^2}\sqrt{\sum_{i=1}^{n} V_i^2}}$$, onde *Ui* e *Vi* são as componentes dos vetores *U* e *V*, respectivamente.

Exemplo:

Seja o seguinte espaço vetorial de 3 dimensões contendo os vetores R e S.

	R	S
X	1	5
Y	2	6
Z	3	1

Logo,

$$\cos(\alpha) = \frac{(1x5) + (2x6) + (3x1)}{\sqrt{1^2 + 2^2 + 3^2} \text{ x } \sqrt{5^2 + 6^2 + 1^2}}$$

$$\cos(\alpha) = \frac{20}{\sqrt{14} \text{ x} \sqrt{62}} \approx 0.678844233$$

$$\alpha = \arccos(0.678844233) \approx 47.246\,°$$

Figura 1

Questão 4 (30 pontos)

Construa um arquivo python chamado *questao4.py.* O arquivo deverá conter as seguintes funções (utilize os mesmos nomes de funções fornecidos no enunciado):

a) Assinatura e funcionalidade: ***matrizFreq(<txt nome arquivo>)***, recebe como entrada um texto contendo um nome de arquivo tipo texto. O arquivo contém uma lista com nome de outros arquivos texto, 1 arquivo por linha, representando os documentos de uma coleção (corpus). Após a obtenção da lista de arquivos do corpus, a função deve processá-los e gerar uma tabela de frequência de tokens do tipo palavras simples, palavras compostas e datas (questão 1, item c). Separadores não estão incluídos. (20 pts).

b) Assinatura e funcionalidade: ***main***, aplicação funcionando via linha de comando. Quando chamado pelo terminal, deve receber como parâmetro de entrada o nome de arquivo contendo a lista de arquivos do corpus, justamente o nome que deve ser repassado para a função ***matrizFreq***, item a). A aplicação deve gerar a tabela de frequência do corpus e exibi-la na tela. (10 pts)

Obs: os arquivos para processamento estão na pasta corpus, disponibilizado como material de prova. Dentro dessa pasta, o arquivo *lstcorpus.txt* contém a lista com os nomes dos arquivos presentes na pasta.

Dica: trate a pasta corpus como um espaço vetorial.

Boa Prova!