Fundação CECIERJ - **Vice-Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação
Professores: Dante Corbucci Filho e Luís Felipe Ignácio Cunha

APX 3 – 1° semestre de 2022

IMPORTANTE: As respostas (programas) deverão ser entregues pela plataforma em um arquivo ZIP contendo todos os arquivos de código fonte (extensão ".py") necessários para que os programas sejam testados. Respostas entregues fora do formato especificado, por exemplo, em arquivos com extensão ".pdf", ".doc" ou outras, não serão corrigidas.

- Serão aceitos apenas soluções escritas na linguagem Python 3. Programas com erro de interpretação não serão corrigidos. Evite problemas utilizando tanto a versão da linguagem de programação (Python 3.X) quanto a IDE (PyCharm) indicadas na Aula 1.
- Quando o enunciado de uma questão inclui especificação de formato de entrada e saída, tal especificação deve ser seguida à risca pelo programa entregue. Atender ao enunciado faz parte da avaliação e da composição da nota final.
- Os exemplos fornecidos nos enunciados das questões correspondem a casos específicos apontados para fins de ilustração e não correspondem ao universo completo de entradas possíveis especificado no enunciado. Os programas entregues devem ser elaborados considerando qualquer caso que siga a especificação e não apenas os exemplos dados. Essa é a prática adotada tanto na elaboração das listas exercícios desta disciplina quanto no mercado de trabalho.
- Faça uso de boas práticas de programação, em especial, na escolha de identificadores de variáveis, subprogramas e comentários no código.
- As respostas deverão ser entregues via atividade específica na Plataforma antes da data final de entrega estabelecida no calendário de entrega de ADs. Não serão aceitas entregas tardias ou substituição de respostas após término do prazo.
- As ADs são um mecanismo de avaliação individual. As soluções podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual. Respostas plagiadas não serão corrigidas.

Boa Avaliação!

1a Questão (3,0 pontos)

Faça um programa, contendo subprogramas, que leia da entrada padrão o nome de um arquivo texto contendo vários pontos que representam um polígono, um ponto por linha. Calcule e escreva o perímetro do polígono formado por estes pontos. Suponha que o primeiro ponto se conecte com o segundo, o segundo ponto se conecte com o terceiro, e assim sucessivamente, e o último ponto no arquivo se conecte com o primeiro ponto. Mostre o conteúdo do arquivo de pontos. Em seguida, caso o arquivo esteja vazio, escreva na saída padrão a mensagem "Arquivo de pontos está vazio!!!". Caso contrário, escreva a mensagem "Perímetro do polígono de pontos contidos no arquivo:" seguida do valor do perímetro calculado, com dupla precisão.

Definição: A distância entre dois pontos (xA, yA) e (xB, yB) é dada pela raiz quadrada da soma do quadrado das diferenças entre as coordenadas dos dois pontos, (xB-xA) e (yB-yA).

Restrição: É proibido manter todos os pontos contidos no arquivo em memória principal, por exemplo em listas.

Exemplo:

Entrada	Saída
Vazio.txt	Conteúdo do Arquivo Vazio.txt: Arquivo de pontos está vazio!!!
UmPonto.txt	Saídas Correpondentes: Conteúdo do Arquivo UmPonto.txt: 50 300 Perímetro do polígono de pontos contidos no arquivo: 0.00
Pontos.txt	Conteúdo do Arquivo Pontos.txt: 5 80 10 40 30 400 350 300 -100 200 10 -10 99 70 30 90 50 350 20 500 Perímetro do polígono de pontos contidos no arquivo: 2459.69

2a Questão (4,0 pontos)

Faça em **um único arquivo .py** as seguintes manipulações para uma entrada:

a) (1,0) Verifique se a entrada possui em cada posição um elemento de 0 até 9. Caso haja algum elemento que não satisfaça essa restrição, então retorne a seguinte mensagem “A entrada {entrada} não está do formato solicitado”, após isso feche o programa. Caso todos os elementos satisfaçam a condição pedida, então converta a entrada no tipo str.

b) (1,0) Determine quantos são os elementos que estão em posições iguais aos seus valores. Assuma que os índices dos elementos vão a partir da posição 1.

c) (2,0) Dada uma lista de entrada, verifique se essa lista está ordenada. Para o caso da lista ordenada, obtenha os elementos que sejam pares na mesma ordem apresentada na entrada.

ATENÇÃO: Você deve fazer um único arquivo .py esta questão. Caso contrário, sua questão pode não ser corrigida ou então não considerada por completo!

Exemplo:

Entrada	Saída
23458	Na entrada 23458 não há elementos com valores iguais as suas posições A entrada 23458 está ordenada A lista dos elementos pares de 23458 em ordem de aparição é ['2', '4', '8']
123 5 34234	A entrada 123 5 34234 não está do formato solicitado
225948261	Na entrada 225948261 há 1 elemento com valor igual a sua posição A entrada 225948261 não está ordenada
1234,2123	A entrada 1234,2123 não está do formato solicitado

3a Questão (3,0 pontos)

Faça em **um único arquivo .py** as seguintes manipulações para uma entrada:

Considere o seguinte problema, desejamos montar grupos de pessoas que sejam de dois tipos, um grupo possui um único elemento ou um grupo possui dois elementos. Uma configuração é uma possível disposição das escolhas dentre todos os grupos de elementos. Por exemplo, caso existam 3 elementos a, b, c, temos as seguintes configurações: {a, b, c} (os três elementos ficam em grupos sozinhos), {a, bc} (a fica sozinho e b e c ficam juntos), {ab, c} (a e b ficam juntos e c fica sozinho), {ac, b} (a e c ficam juntos e b fica sozinho). Ou seja, para o caso de 3 elementos, temos 4 configurações possíveis. Neste item, você deve calcular o número de configurações possíveis para qualquer inteiro maior do que zero dado de entrada.

Dica: Para fazer essa computação, imagine que há todas as configurações para quando há n-1 elementos, nelas você vai adicionar um novo elemento (o n-ésimo elemento). Esse novo elemento pode ficar sozinho (gerando configurações em que os n-1 elementos anteriores não alteram as configurações) ou então esse n-ésimo elemento pode escolher 1 dos n-1 elementos existentes para fazer um par (dessa forma, sobram n-2 elementos para calcularmos as computações).

ATENÇÃO: Você deve fazer um único arquivo .py esta questão. Caso contrário, sua questão pode não ser corrigida ou então não considerada por completo!

a) (1,5) Implemente essa estratégia usando iteratividade.

b) (1,5) Implemente essa estratégia usando recursividade.

Exemplo:

Entrada	Saída
6	Recursivamente, temos 76 possibilidades de jogos Iterativamente, temos 76 possibilidades de jogos.
12	Recursivamente, temos 140152 possibilidades de jogos Iterativamente, temos 140152 possibilidades de jogos.

Boa Avaliação!