Ifes Campus Serra
BSI – Bacharelado de Sistemas de Informação
Programação II

Avaliação 3: Tipos Abstratos de Dados

Regras para a prova: utilize apenas os comandos python vistos em sala de aula // para manipulação de arquivos, utilizar apenas open, close, readline, write // não utilizar expressões de lista // não utilizar dados com tipos // na construção da prova, utilizar apenas os arquivos de dados fornecidos pelo Professor, SEM modificações // para construir as respostas, utilize os nomes de arquivos .py e nomes de funções conforme pedidos nos enunciados // todas as figuras exibem apenas *exemplos* para auxiliar na interpretação do enunciado // celulares, pendrives, não são permitidos durante a prova.

Em outro momento, fora da aplicação, o Professor pode solicitar que o aluno explique a solução da prova.

A prova é indivual e sem acesso à Internet.

Questão Única

Construa o tad ponto (**tadponto.py**) obdecendo as seguintes especificações: um tad ponto representa um ponto no plano por meio das coordenadas x e y. (estimado 20 minutos)

Interface do tad ponto:

a) **def new_ponto(x, y)**: retorna a estrutura selecionada pelo desenvolvedor para representar o tad ponto e seus dados.

b) **def get_x(tadponto)**, **def get_y(tadponto**): retornam os valores das coordenadas x e y respectivamente.

c) **def distancia(tadpontoA, tadpontoB)**: retorna distância eulcidiana entre os pontos representados pelos 2 parâmetros tad ponto.

Construa o tad estacionamento (tadest.py) obdecendo as seguintes especificações: um tad estacionamento carrega as informações mostradas na figura 1. (estimado 40 minutos)

O tad estacionamento precisa importar o tad ponto e fazer uso apenas da sua respectiva interface.

Interface do tad estacionamento:

a) **def new_est(es, di, empresa, email)**: retorna o dicionario exibido na figura 2, contendo os dados de um estacionamento (figura 1). **es** e **di** são tad pontos que representam os cantos do retângulo que é o espaço físico do estacionamento (ver figura 1).

b) **def get_empresa(tadest):** retorna o valor da empresa dona do estacionamento.

c) **def get_email(tadest):** retorna o valor do e-mail de contato da empresa dona do estacionamento.

d) **def get_num_carros(tadest)**: retorna o valor atual da quantidade de carros no estacionamento.

e) **def set_num_carros(tadest, num):** atualiza o valor da quantidade de carros no estacionamento.

f) **def area(tadest):** retorna o valor da área do estacionamento. Apenas funções do tad ponto devem ser utilizadas para o cálculo. Lembre que os pontos **ds** e **ei** podem ser criados a partir dos pontos **es** e **di**.

g) **def perimetro(tadest)**): retorna o valor do perímetro do estacionamento. Apenas funções do tad ponto devem ser utilizadas para o cálculo.

h) **def pertence(tadest, tadponto)**: retorna True se o tad ponto pertence ao espaço do estacionamento, retorna False caso contrário.

- -

Construa a aplicação função main (resposta.py). A aplicação importa os tads anteriores e resolve o seguinte problema (utilizando **apenas** as interfaces dos tads). (estimado 40 minutos)

i. Crie uma lista de estacionamentos a partir do arquivo **bdestacionamentos.txt**. Cada linha do arquivo contem os dados de um estacionamento. A cada linha lida, um tad estacionamento deve ser criado e armazenado na lista.

ii. Abra o arquivo **bdveiculos.txt** e faça a leitura linha a linha. Em cada linha deste arquivo consta as coordenada x e y (latitude, longitude) de um veiculo no plano.

iii. Para cada linha linha lida do arquivo **bdveiculos.txt**, crie um tadponto (representando um veículo) e teste se este pertence a um dos estacionamentos da lista criada em i). Se pertencer, atualizar a quantidade de carros desse estacionamento.

iv. Percorrer a lista criada em i) e exibir o nome da empresa, o e-mail, a área, o perimetro e a quantidade de carros **da empresa que possui o maior número de carros em seu estacionamento**.

Figura 1

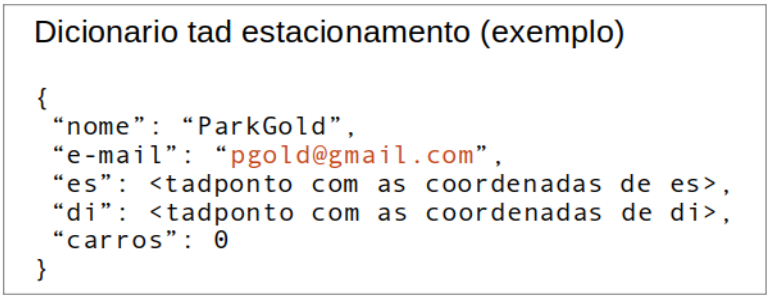

Figura 2

Boa prova!