Faculdade
IMPACTA

ESTRUTURA DE DADOS

ATIVIDADE CONTINUA Nº 2 – AC 02

PROFESSOR: JORGE CARLOS VALVERDE REBAZA

SEMESTRE: 2020 - I

Instruções

A seguir são apresentadas duas (2) perguntas. Umas valendo 4 pontos e outra valendo 6 pontos. Siga as instruções abaixo:

1. Desenvolver de maneira individual ou, no máximo, em duplas. Não é permitido, sob nenhum suposto, grupos de mais de 2 alunos.
    a. Grupos com mais de 2 alunos terão uma **punição de 4 pontos**.
    b. Grupos de 1 aluno ou de 2 alunos serão tratados de igual maneira. Não haverá distinção. Então, é muito recomendável trabalhar em duplas.

2. Para cada pergunta deverá ser enviado apenas um arquivo Python (.py). Isto é, são dois (2) perguntas pelo que deverão ser enviados como respostas, apenas, dois (2) arquivos Python.
    a. Arquivos em outro formato que não seja Python (.py) terão uma **punição de 3 pontos**.
    b. Cada arquivo deverá ter no seu cabeçalho os seguintes dados como comentário: nome da disciplina e da turma, número e título da pergunta, Nome COMPLETO dos alunos que fazem parte do grupo, data de envio, nome do professor. Arquivos que tenham algum desses dados faltando ou incompleto terão uma **punição de 2 pontos**.
    c. Arquivos com soluções parecidas o que tenham mínima suspeita de cópia/plágio, terão **punição de 5 pontos**. O julgamento da suspeita de cópia fica apenas sob os critérios do professor. Então, trabalhe apenas com seu colega de equipe.
    d. É OBRIGATÓRIO, comentar seu código para facilitar o entendimento da lógica do programa. Soluções que careçam de código e/ou sejam insuficientemente comentados, terão uma **punição de 1 ponto**.

3. Apesar de trabalhar em grupos, CADA membro do grupo tem a OBRIGAÇÃO de enviar os arquivos da solução de maneira INDIVIDUAL via classroom.
    a. Aluno que não envie sua solução via classroom não terá o respectivo arquivo avaliado e terá, por conseguinte, nota zero (0) na respectiva

pergunta. Não adianta que o colega de grupo tenha feito o envio, cada um tem que fazer o enviar obrigatoriamente.

4. O envio dos arquivos de solução é exclusivamente via classroom na data devidamente informada.
    a. Arquivos enviados por email o outro médio NÃO serão considerados e terá, por conseguinte, nota zero (0) na respectiva pergunta.
    b. Envios fora da data informada terão uma **punição de 1 ponto por dia**.
    c. O classroom permite o envio de apenas um (1) único envio, então, assegure-se de carregar seus dois (2) arquivos de respostas antes de enviar (pode enviar 2 arquivos .py ou usar algum compressor tipo winzip ou winrar e enviar uma pasta compressa em formato .zip ou .rar).

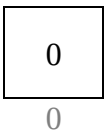
Pergunta 1: Likes de Postagens em Redes Sociais (4 pontos)

Uma empresa precisa ter um registro dos likes que recebem os diferentes posts publicados pela sua agência publicitária. Para isso, solicitam a você construir um programa em Python que, mediante o uso de um TAD, consiga administrar as seguintes operações:

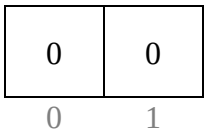
1) **Registrar um novo post:** Um post é representado como um elemento de uma lista. O índice da lista indica o número de post -1. Por exemplo, o primeiro post (postagem número 1) corresponde ao índice zero (0) da lista. O valor armazenado na posição indicada pelo índice corresponde ao número de likes da postagem. Toda postagem começa com zero likes.

    Exemplo:

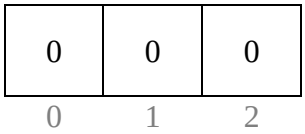
    - Registrar um post (primeira postagem):



    - Registrar um novo post (segunda postagem):



    - Registrar um novo post (terceira postagem):



2) **Registrar um número específico de likes a uma postagem:** O número de likes é acumulativo, isto é, se a postagem tem 0 likes e recebe 5 likes, ela terá 0 + 5 = 5 likes. Se logo, essa mesma postagem recebe 7 likes, terá 5 + 7 = 12 likes.

3) **Política de atualização e benefícios de likes:** Cada vez que uma postagem recebe um número de likes específico, automaticamente as postagens vizinhas (com índices anterior e posterior) sofrerão também uma atualização seguindo a seguinte política:
   - Se a postagem original recebe entre 1 e 10 likes, então seus vizinhos recebem 1 like de bônus.
   - Se a postagem original recebe mais de 10 likes, então seus vizinhos recebem a metade de likes recebidas pela postagem original.

Exemplo:

- Atribuir 5 likes à postagem com índice 0 (seu único vizinho é o índice 1)

5	1	0
0	1	2

- Atribuir 12 likes à postagem com índice 1 (seus vizinhos são os índices 0 e 2). Nesse caso, 6 likes são atribuídos como bônus aos seus vizinhos.

11	13	6
0	1	2

4) **Erro ao atribuir likes.** Se por engano há uma tentativa de atribuir likes a uma postagem não existente, além de mostrar uma mensagem de erro, esses likes devem ser atribuídos à uma postagem aleatória, sem beneficiar (entregar bônus) aos vizinhos dessa postagem.

   Exemplo:
   - Atribuir 8 likes à postagem com índice 5. Como neste caso essa postagem não existe, é escolhida uma postagem aleatória. Supondo que a postagem aleatória corresponde ao índice 1, então ela recebe os likes, mas seus vizinhos não recebem bônus.

11	21	6
0	1	2

5) **Busca das postagens com mais likes:** Cada vez que requerido, é necessário apresentar o top-3 de postagens com mais likes.

O programa em Python que você irá construir deverá cumprir as políticas definidas acima de maneira dinâmica e de acordo às solicitações que o usuário faça pela consola. Lembre-se, a gestão da publicação de postagens e likes deve ser feito através de um TAD chamado GestorPostagens o qual gerenciará as políticas acima descritas mediante o usa de uma estrutura de dados lista.

A interação do usuário com o programa deverá ser feita mediante um menu simples o qual, além de mostrar as opções que o usuário pode realizar, deverá oferecer a atualização dos valores da lista de postagens.

Exemplo:

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top posts com mais likes

Escreva uma opção: **1**

O post nro. 1 tem sido criado.

A lista de postagens é a seguinte:

Índice: 0

Likes: 0

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **1**

O post nro. 2 tem sido criado.

A lista de postagens é a seguinte:

Índice: 0 1

Likes: 0 0

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **1**

O post nro. 3 tem sido criado.

A lista de postagens é a seguinte:

Índice: 0 1 2

Likes: 0 0 0

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **2**

Ingresse o número do post ao qual você quer dar likes: 1

Ingresse o número de likes que você quer atribuir: 5

Índice: 0 1 2

Likes: 5 1 0

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **2**

Ingresse o número do post ao qual você quer dar likes: 2

Ingresse o número de likes que você quer atribuir: 12

Índice: 0 1 2

Likes: 11 13 6

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **2**

Ingresse o número do post ao qual você quer dar likes: 6

Ingresse o número de likes que você quer atribuir: 8

Índice: 0 1 2

Likes: 11 21 6

Aperte R para voltar

Menu Sistema Gestor de Postagens

1) Criar um post
2) Dar likes a um post
3) Top 3 posts com mais likes

Escreva uma opção: **3**

O top-3 posts com mais likes são:

Top: 1 2 3

Índice: 1 0 2

Likes: 21 11 6

Aperte R para voltar

Pergunta 2: Rover Curiosity (6 pontos)

A missão do robô *Rover Curiosity* chegou à cratera Gale de Marte o 6 de agosto de 2012. Desde essa data, o robô começou a explorar o território de Marte. O robô tem, apenas, um braço e uma câmera para fazer a exploração. O *Curiosity* chega a correr a uma velocidade de até 90 m/h, tem uma massa de 899 kg, uma altura de 2,2m, uma largura de 2,7m e um cumprimento de 3m. Seu braço se estica até 3m. Além da câmera, conta também com uma bateria solar.

O *Curiosity* pega elementos que encontra no solo de Marte, os avalia com sua câmera e os salva em até 2 depósitos diferentes. Se o que ele encontra é avaliado como uma rocha, então tal rocha é salva num depósito em formato de lista. Se o que for encontrado trata-se de partes de lixo deixado por explorações passadas, então esse lixo é salvo num outro depósito em formato de pilha.

O *Curiosity* salva rochas de ate 2,5kg de peso e de até 0,74m de diâmetro. Rochas maiores a esse peso e dimensão, são desconsideradas. Rochas podem ser dos tipos 1, 2 e 3. Se a capacidade de armazenamento do *Curiosity* fica comprometida, o robô avalia a quantidade de rochas em função do seu tipo e, rejeita (elimina) as rochas necessárias até ter uma quantidade mais ou menos equilibrada de tipos de rochas.

Ao mesmo tempo, o robô salva lixo espacial de até 2,5kg de peso e de até 0,3m de diâmetro. Lixo espacial que ultrapasse esse peso e dimensão, é desconsiderado. Lixo espacial pode ser de tipo metálico e não metálico. O robô equilibra sua pilha de lixo espacial de maneira que, antes de armazenar um determinado tipo de lixo, avalia qual tipo de lixo está no topo da pilha. Se no topo houver lixo metálico, então o novo elemento a ser empilhado deve ser do tipo não metálico, e vice versa. Caso essa condição não possa ser cumprida, o robô ira desempilhar até ter, no topo da sua pilha, o tipo de lixo adequado para salvar o novo lixo que pretende salvar.

O *Curiosity*, consegue armazenar até 70kg (somando rochas e lixo) durante uma missão de exploração. Quando a capacidade de armazenamento se vê comprometida, o robô avalia se o equilíbrio de tipos de rochas na sua lista de rochas é o adequado. Se a lista não estiver equilibrada, então elimina rochas conforme a política descrita acima. Caso a lista estiver equilibrada e não seja possível inserir mais nenhum elemento (rocha ou lixo) que esteja em avaliação, o robô desconsidera esse elemento e continua a exploração de um novo elemento. Esse processo é repetido até terminar de explorar uma determinada região de Marte.

Após o término de uma exploração, o robô precisa voltar até uma estação espacial para liberar os elementos armazenada e iniciar com uma nova exploração.

Requerer-se que você construa um sistema em Python que, usando TADs, simule a exploração do *Curiosity* em Marte. Assim, construa um TAD Rocha, um TAD LixoEspacial, e um TAD Curiosity. Para sua simulação, crie uma lista de tamanho $30 \leq N \leq 120$ para o TAD Rocha e uma segunda lista para o TAD LixoEspacial, também de tamanho N. O valor de N é gerado aleatoriamente. Preencha os atributos de cada elemento dessas duas listas com valores aleatórios. Considere que as rochas em Marte têm pesos que vão desde 0,5kg até 14,2kg; enquanto há registros de que o lixo espacial em Marte deve pesar entre 1,12kg e 8,55Kg.

Com essas duas listas criadas, você irá percorrê-las ao mesmo tempo e irá inserir no *Curiosity*, uma rocha e um lixo por vez, seguindo as políticas descritas acima. Considere que ambas listas correspondem ao total de elementos a serem explorados pelo robô, isto é, percorrer ambas listas corresponde a uma exploração realizada pelo *Curiosity*.

Mostre mensagens que permitam fazer seguimento do passo a passo das ações do *Curiosity*. Quando o armazenamento do *Curiosity* estiver totalmente preenchido, ou simplesmente não houver mais rochas ou lixos a serem armazenados. Você deverá mostrar na tela o reporte completo da lista e pilha do *Curiosity* e a mensagem dizendo "Curiosity terminou exploração. Liberando espaço de armazenamento".

Fazer a simulação para três (3) explorações do *Curiosity*.