INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Serra

Ministério
da Educação

Programação 2
Trabalho

O aplicativo de paquera que você desenvolveu fez um enorme sucesso, sendo utilizado por milhões de usuários em todo o mundo (se são usuários verdadeiros ou *fakes*, jamais saberemos). Entretanto, uma queda de energia inesperada fez você perceber que não estava salvando seus dados apropriadamente.

O sistema armazena as informações através de dois dicionários: um para os dados dos usuários e um para as conexões entre eles. A chave de todos eles é o ***login*** do usuário.

O conteúdo do primeiro dicionário é uma tupla com ***nome***, ***cidade*** e ***data de nascimento*** de cada usuário:

```
              #LOGIN   : ( NOME    ,  CIDADE ,  NASCIMENTO )
usuarios = { "pkchu" : ("Pikachu", "Kanto", (4,12,2000)),
             "drgnt" : ("Dragonite", "Kanto", (1,10,2002)),
             "jpuff" : ("Jigglypuff", "Kanto", (15,2,1988)),
             "blssy" : ("Blissey", "Johto", (2,3,2000)),
             "trntr" : ("Tyranitar", "Johto", (28,6,2001)) }
```

O segundo armazena uma tupla com três listas: dos perfis que ele tem interesse, dos perfis que demonstraram algum interesse por ele, e dos perfis em que há interesse mútuo das duas partes.

```
              #LOGIN   : ( GOSTEI, GOSTOU, MÚTUOS )
conexoes = { "pkchu" : (["jpuff", "blssy"], [], []),
             "drgnt" : ([], ["jpuff"], ["blssy"]),
             "jpuff" : (["drgnt"], ["pkchu"], ["trntr"]),
             "blssy" : (["trntr"], ["pkchu"], ["drgnt"]),
             "trntr" : ([], ["blssy"], ["jpuff"]) }
```

Note que não pode haver inconsistências entre as três listas. Não pode haver, por exemplo, algum usuário Y que apareça em mais de uma lista de um mesmo usuário X.

Se o usuário X gostou de Y, mas Y não demonstrou interesse por X até o momento, então Y fica armazenado na lista dos perfis que X gostou e X fica na lista dos que demonstraram interesse por Y. Entretanto, se Y vier a demonstrar interesse por X, ***as listas de conexões de X e de Y devem ser atualizadas:***

- Y sai da lista de usuários que X gostou, e passa para a lista de interesses mútuos de X
- X sai da lista dos interessados por Y, e vai para a lista de interesses mútuos de Y.

No exemplo, o usuário chamado “Jigglypuff” já demonstrou interesse por “Dragonite”, mas o interesse ainda não foi recíproco. Caso, em algum momento, isso aconteça, o dicionário deverá ficar assim:

```
conexoes = { "pkchu" : (["jpuff", "blssy"], [], []),
             "drgnt" : ([], [], ["blssy", "jpuff"]),
             "jpuff" : ([], ["pkchu"], ["trntr", "drgnt"]),
             "blssy" : (["trntr"], ["pkchu"], ["drgnt"]),
             "trntr" : ([], ["blssy"], ["jpuff"]) }
```

O sistema salva esses dicionários num único arquivo binário chamado “***backup.bin***”, nesta mesma ordem: o dicionário de usuários e depois o dicionário de conexões.

Além disso, por precaução, o sistema salva também uma lista com o histórico de todos as tentativas de conexões realizadas pelos usuários. Sempre que um usuário **X** indica que gostou de um usuário **Y**, é armazenada nesta lista uma tupla com os logins de **X** e **Y**, nesta ordem.

PARTE 1) RECUPERANDO OS DADOS PERDIDOS COM A QUEDA DE ENERGIA

A queda de energia causou um sério problema no seu sistema. Isso porque quase tudo estava sendo atualizado em disco, exceto as conexões. Por conta disso, o dicionário de usuários está corretamente atualizado no arquivo. Mas não se sabe a última vez em que o dicionário de conexões e a lista com o histórico das todas as tentativas de conexões foram atualizados.

Sua primeira missão é criar um programa chamado “gerarConexoes.py” que vai percorrer o histórico com as tentativas de conexões, atualizando o dicionário de conexões com as informações que não estiverem lá.

Por exemplo, se o dicionário estiver com os dados a seguir:

```
             #LOGIN  : ( GOSTOU,  MÚTUOS )
conexoes = { "pkchu" : (["blssy"], ["jpuff"]),
             "drgnt" : (["jpuff"], []),
             "jpuff" : ([], ["pkchu", "trntr"]),
             "blssy" : (["drgnt"], ["trntr"]),
             "trntr" : ([], ["blssy", "jpuff"] ) }
```

Se existir no histórico uma tupla `("blssy", "trntr")`, nada precisa ser feito. Isso porque o dicionário acima já reflete que o primeiro usuário demonstrou interesse no segundo. Entretanto, se existir no histórico uma tupla `("blssy", "pkchu")`, isso significa que o dicionário precisa ser atualizado apropriadamente com essa informação.

O programa deve criar um novo arquivo binário chamado “***dados.bin***” contendo, nesta ordem: o dicionário com os usuários, e o dicionário atualizado pelo backup com todas as conexões realizadas.

PARTE 2) ESTATÍSTICAS

Por fim, já com os dados atualizados, você quer dar ainda mais destaque no aplicativo para os usuários que estão fazendo mais sucesso (sim, o mundo nem sempre é justo com todos e você só está preocupado em ganhar mais dinheiro).

Para isso, você vai criar outro programa chamado “***estatisticas.py***”, que vai ler os dicionários gerados pelo arquivo “***dados.bin***” e criar uma lista contendo apenas os logins dos usuários (nada além disso, para não

sobrecarregar a memória do computador). Em seguida, você irá ordenar esta lista de logins segundo os seguintes critérios:

1. Em ordem alfabética pela cidade do usuário.
2. Em caso de empate, em ordem *decrescente* pelos usuários que tenham despertado mais interesse nos demais (somando todos os usuários que gostaram dele, tendo ou não sido correspondidos).

Crie um arquivo texto chamado "***top.txt***" e salve nesse arquivo o usuário mais popular de cada cidade de acordo com a ordem estabelecida acima. Em cada linha, o arquivo vai contar o nome da cidade e o nome completo do usuário, separados por um espaço em branco. Exemplo:

```
Belem/PA Madalena Lopes
Belo Horizonte/MG Francisca Teixeira
Brasilia/DF Miguel Marques
Campinas/SP Tiago Correia
Campo Grande/MS Marcos Rodrigues
Curitiba/PR Miguel Ferreira
Duque de Caxias/RJ Bruno Pereira
Fortaleza/CE Juliana Pinto
Goiania/GO Paula Ferreira
Guarulhos/SP Carolina Cruz
Maceio/AL Catarina Pereira
Manaus/AM Daniela Guimaraes
Natal/RN Sabrina Gomes
Porto Alegre/RS Francisco Souza
Recife/PE Tomas Costa
Rio de Janeiro/RJ Bruno Santos
Salvador/BA Rodrigo Gama
Sao Goncalo/RJ Tomas Santos
Sao Luis/MA Ana Veiga
Sao Paulo/SP Denise Mendes
```

Observações

- O trabalho vale 40 pontos, pode ser feito em grupo de até 3 integrantes.
- Trabalhos considerados plágio terão nota 0 para quem copiou e para quem forneceu o trabalho. Além disso, serão enviados para o Conselho de Ética.
- O código deve ser feito em Python3. Deve ser enviado um arquivo compactado contendo apenas os códigos fontes (*.py) do programa. Não inclua arquivos de entrada e/ou de saída na submissão do trabalho.
- Os nomes dos integrantes do grupo devem aparecer no nome do arquivo compactado e comentado no início do código fonte principal.
- Trabalhos entregues após o prazo, com erro de execução, com formato de saída incorreto, ou que não compilarem terão nota 0.
- O trabalho deve ser enviado na sala da disciplina do AVA.
- Em caso de dúvidas na especificação do trabalho ou no próprio trabalho, contate-me em sala.