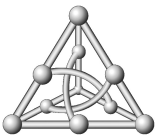


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Hashtag

Trabalho 2

Algoritmos e Programação I

1 Descrição

Achadas em certas áreas das Américas do Sul e Central, próximo a rios e lagos, a **capivara** (*Hydrochoerus hydrochaeris*), também chamada de carpincho e capincho, é o maior roedor do mundo. Quando a esquadra de Pedro Álvares Cabral chegou ao Brasil em 1500, os indígenas locais já domesticavam este animal. Alimenta-se de capins e ervas, daí, a etimologia de seu nome: capivara procede do termo tupi *kapi'wara*, que significa "comedor de capim". Já capincho vem do castelhano platino *capincho*. No Rio Grande do Sul, é também conhecida por capinga.

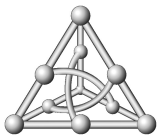
Esse é um bicho amado em diversas culturas, como pode ser visto na Figura 1. No Brasil, as políticas de inclusão digital têm alcançado até mesmo as capivaras, de modo que elas passaram a se comunicar por uma certa rede de microblogs que utiliza as conhecidas *hashtags*[1]. Uma preocupação emergente é identificar quais são os temas mais discutidos entre elas, para assim planejar ações adequadas à garantia de seu bem-estar.

Figura 1: (a) capivara na FACOM; (b) capivara chilena; (c) capivaras japonesas e (d) nota de 1 pila capybariano.

Sua tarefa neste trabalho é escrever um programa que receba um conjunto de postagens, que podem incluir até uma[2] *hashtag*, realizando as seguintes operações sobre esse conjunto de dados:

1. Inserção (cadastro/inclusão) de uma postagem;
2. Listagem de todas *hashtags*, na ordem em que apareceram pela primeira vez, incluindo a contagem de quantas vezes foram utilizadas.
3. Impressão de todas postagens contendo uma determinada *hashtag*;

[1] http://pt.wikipedia.org/wiki/Hashtag
[2] **Sempre** que aparecer o caractere #, este indica o início de uma *hashtag*



2 Entrada e saída

A entrada contém vários casos de teste, cada um representando um conjunto de dados sobre o qual iremos trabalhar. A primeira linha contém um inteiro $k > 0$ que se refere ao número de casos de teste. Após esse número, uma linha em branco será digitada, e a seguir, são apresentadas as informações dos k casos de teste, em sequência. Após finalizar um caso de teste, o usuário sempre digitará uma linha em branco, apenas para visualizarmos melhor os dados.

Para cada caso de teste, são dadas várias operações entre as apresentadas na seção anterior, uma por linha. A seguir iremos detalhar a forma com que cada operação é lida (entrada) e quais informações devem ser impressas (saída). Para todas operações, considere que uma hashtag tem o formato `#descritor`, sendo que o descritor contém apenas caracteres de **a** a **z**, minúsculos ou maiúsculos e sem acentos. Seu programa deve considerar *hashtags* independente das letras maiúsculas ou minúsculas nela, então, por exemplo, `#palAvrA` e `#PALAvra` devem ser consideradas *hashtags* iguais.

2.1 Inserção

A entrada para a inserção é dada no formato `I postagem`, onde `postagem` equivale a uma frase qualquer sem acentos, podendo ter zero ou qualquer quantidade de *hashtags* em qualquer posição da frase. Essa operação não deve gerar nenhuma saída, ou seja, não imprima nada, apenas inclua a nova postagem na lista de postagens. **Considere que** uma *hashtag* sempre aparecerá no início da linha ou após um espaço, mas após uma *hashtag* podem aparecer outros símbolos (p. ex. em `#sol...` os pontos não fazem parte da *hashtag*). **Considere que** uma mesma *hashtag* não aparece duas vezes no mesmo post.

- Exemplo de entrada da operação:

```
I Sempre que a #chuva cai eu fico molhado.
```

2.2 Listagem

Para pedir uma listagem de *hashtags* com suas respectivas contagens, o usuário deve apenas digitar `L` em uma linha. O seu programa então deve imprimir **em letras minúsculas** uma lista de todas as *hashtags* inseridas até o momento, cada uma seguida da quantidade de vezes que ela aparece nas postagens e de uma quebra de linha.

- Exemplo de entrada da operação:

```
L
```

- Exemplo de saída da operação (supondo que várias postagens já foram inseridas):

```
#chuva 3
#triste 1
#olimpiadas 18
```

2.3 Impressão

A operação de impressão de todas as postagens com uma determinada *hashtag* é solicitada ao digitar `P #descritor`, onde `#descritor` refere-se à *hashtag* em questão. Seu programa deve imprimir todas as postagens incluídas até o momento que contenham a *hashtag* solicitada, uma postagem por linha. Observe que, por exemplo, `P #chUVA` e `P #chuva` devem produzir o mesmo resultado.

- Exemplo de entrada da operação:

```
P #chuva
```

- Exemplo de saída da operação (supondo que várias postagens já foram inseridas):

```
Sempre que a #ChUva cai eu fico molhado.
Essa #chuva nao para.
Adoro #CHUVA.
```

2.4 Finalização

Quando o usuário digitar em uma linha `F`, significa o fim do caso de teste atual. Nesse caso, imprima apenas uma linha em branco. Sugere-se aqui zerar contadores e etc, caso ache necessário.

- Exemplo de entrada da operação:

```
F
```

- Exemplo de saída da operação:

3 Exemplo de entrada

```
2

I Hoje eh um bom #dia.
I Outra vez perdi a hora.
I Minha #capimae eh legal.
I #choveu ontem...
I Cade o #sol?
L
P #sol
I Preciso de #sol...
I Nao faz #Sol ha uma semana
L
P #SOL
F

I Sempre que a #chuva cai eu fico molhado.
I Quando comecam as #olimpiadas?
L
I A chuva me deixa #triste
I Corri feito louco de um jacare... :-(
I Nao vai ter #olimpiadas!
I Essa #chuva nao para.
I Adoro #CHUVA!!!
I Esse pessoal da #facom so pisa na minha grama, como eh que vou almocar?
I E da-lhe #chuva fria :-\
I O bom da chuva eh que a #grama fica mais gostosa.
L
P #chuva
F
```

4 Exemplo de saída

```
#dia 1
#capimae 1
#choveu 1
#sol 1
Cade o #sol?
#dia 1
#capimae 1
#choveu 1
#sol 3
Cade o #sol?
Preciso de #sol...
Nao faz #Sol ha uma semana

#chuva 1
#olimpiadas 1
#chuva 4
#olimpiadas 2
#triste 1
#facom 1
#grama 1
Sempre que a #chuva cai eu fico molhado.
Essa #chuva nao para.
Adoro #CHUVA!!!
E da-lhe #chuva fria :-\
```

5 Exigências

Você deve utilizar apenas listas para armazenar os dados, portanto NÃO use outras estruturas como *dict* ou *set* por exemplo, caso contrário sua nota será ZERO. Você é livre para usar todas as funções disponíveis nativamente para listas e strings. Se estiver com dúvida se é permitido utilizar um *import* específico, pergunte ao professor.

6 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

   Seu trabalho deve ter um cabeçalho com o seguinte formato:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#
# Nome do(a) estudante
# Trabalho 2
# Professor(a): Nome do(a) professor(a)
#
```

2. **Forma de entrega**

   A entrega será realizada diretamente no Ambiente Virtual de Aprendizagem - UFMS (https://ava.ufms.br), na nossa disciplina de Algoritmos e Programação I. Um fórum de discussão deste trabalho já se encontra aberto. Para entrega do trabalho, vá até o tópico "Trabalhos", e escolha "T2 - Entrega". Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 59 minutos** do dia **07 de junho de 2020**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

3. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

4. **Erros**

   Trabalhos com erros de execução (*Runtime error*) receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de execução.

5. **O que entregar?**

   Você deve entregar um único arquivo contendo APENAS o seu programa fonte com o mesmo nome de seu nome e um sobrenome, como por exemplo, `fulano_silva.py`.

6. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10, você pode assumir que quem executar seu programa nunca irá digitar um valor fora desse intervalo.

7. **Arquivo com o programa fonte**

   Seu arquivo contendo o código Python deve estar bem organizado. Um programa em qualquer linguagem tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso.

8. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO. Estudante que se envolver em DOIS CASOS DE PLÁGIO estará automaticamente REPROVADO na disciplina.