INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados

1º semestre | 2014/2015

Enunciado do 2º Projecto

Data de entrega: Sexta-feira, 1 de Dezembro de 2014 (23h59)

1. Introdução

Neste projecto pretende-se desenvolver um programa em linguagem C que permita o processamento de mensagens (denominadas "bitaites") emitidas por um conjunto de utilizadores da rede social *Bitaiter*. Cada bitaite tem um emissor e é classificado com uma hashtag. Associada a cada hashtag está uma lista dos bitaites emitidos com essa hashtag. Os utilizadores poderão listar os bitaites associados a uma hashtag, por ordem de emissão. O criador de uma hashtag (i.e., o emissor do primeiro bitaite classificado com essa hashtag) pode eliminar a hashtag, bem como a lista de bitaites a ela associada. O programa deverá permitir o registo de bitaites por parte de qualquer utilizador, a listagem ordenada dos bitaites associados a hashtags e a eliminação de hashtags e correspondentes listas de bitaites.

O programa deverá processar um conjunto de linhas, cada uma começando com um dos comandos listados na secção seguinte, e que indicam as operações a executar. Cada utilizador é identificado por um nome que consiste numa sequência de 1 a 20 caracteres não brancos. Os bitaites são sequências de 1 a 200 caracteres (incluindo espaços). As hashtags são sequências de 1 a 40 caracteres não-brancos, sendo o primeiro deles o caractere '#'.

2. Dados de entrada

Os dados de entrada deverão ser recebidos pelo programa através do *standard input*. O programa deverá começar por ler o número total máximo de hashtags (**`maxHashtag`**) que pode existir em qualquer momento da execução do programa. De seguida, o programa deverá processar uma sequência de linhas que indicam as operações a executar. Cada linha contém um único comando de entre os comandos definidos de seguida:

- O comando **`send emissor hashtag bitaite`** permite registar a mensagem **`bitaite`** enviada pelo utilizador **`emissor`** e associa-la à hashtag **`hashtag`**. Este comando não tem qualquer output.
  Exemplo:
  ```
  send @Philae2014 #CometLanding Finally! I'm stretching my legs after more than 10 years. Landing gear deployed!
  ```
  Dica: Para a leitura da mensagem, ver a função **`LeLinha`** dada nas teóricas ou a função **`fgets`**.

- O comando **`list hashtag`** imprime todas as mensagens associadas à hashtag **`hashtag`** (ou **`NULL`** caso não exista). A lista de bitaites deve ser impressa pela ordem com que foram emitidos. Se a hashtag não existir, o comando deverá escrever **`NULL`**. Ver detalhes do output na secção seguinte.
  Exemplo:
  ```
  list #CometLanding
  ```

- O comando **`close emissor hashtag`** permite ao utilizador **`emissor`** apagar o registo da hashtag **`hashtag`**, bem como da lista de bitaites associada, desde que essa hashtag tenha sido criada por **`emissor`**. Se o comando for emitido por um outro utilizador que não o criador da **`hashtag`**, ele não terá qualquer efeito.
  Exemplo:
  `close @giovmusic #onMars`

- O comando **`quit`** apaga todas as hashtags e listas de bitaites associadas, saindo em seguida do programa. Não tem output.
  Exemplo:
  `quit`

3. Dados de saída

Todos os dados de saída deverão ser escritos no *standard output*. Apenas o comando **`list`** implica a escrita de informação. Se não existir qualquer mensagem a ser escrita, o programa deverá escrever a palavra **`NULL`** (seguida do caractere '**`\n`**') no *standard output*. Cada mensagem **`bitaite`** emitida pelo utilizador **`emissor`** com hashtag **`hashtag`** deverá ser apresentada numa linha distinta sob a forma:

`emissor hashtag : bitaite`

Os únicos caracteres brancos em cada linha, para além dos espaços que fazem parte da mensagem **`bitaite`**, são um espaço de separação entre o nome do emissor e a hashtag, entre a hashtag e o caractere ':', entre este e o bitaite, e o caractere mudança de linha '\n' imediatamente após o bitaite.
Exemplo:
`@Wallace #Vegs : Veg bad. Veg bad. Veg bad. Say no to carrots, cabbage and cauliflower.`

Pode assumir que os dados de entrada obedecem sempre às regras descritas na secção anterior. Neste contexto, o programa não precisa de fazer validação de dados de entrada nem produzir qualquer mensagem de erro.

4. Exemplo

4.1. Dados de entrada

```
5
send @Tintin #MoonLanding L'instant est solennel... La porte extérieure
tourne lentement sur ces gonds et...
send @Tintin #MoonLanding Ooooh!... Quel spectacle hallucinant!
send @Tintin #MoonLanding Je descends à présent les échelons qui courent
le long de la fusée...
send @Tintin #MoonLanding Ça y est!... J'ai fait quelques pas!...
send @Tintin #MoonLanding Pour la première fois sans doute dans
l'histoire de l'humanité, ON A MARCHÉ SUR LA LUNE!
send @Armstrong #MoonLanding Okay. I'm going to step off the LM now.
close @Tintin #MoonLanding
send @Armstrong #MoonLanding That's one small step for man; one giant
leap for mankind.
send @Armstrong #MoonLanding Yes, the surface is fine and powdery. I can
kick it up loosely with my toe.
send @Armstrong #MoonLanding Ah... There seems to be no difficulty in
moving around - as we suspected.
close @Tintin #MoonLanding
send @Wallace #Cheese Gromit, that's it! Cheese! We'll go somewhere where
there's cheese!
send @Wallace #Cheese Everybody knows the moon's made of cheese.
list #Gorgonzola
list #MoonLanding
close @Armstrong #MoonLanding
send @Philae2014 #CometLanding Thank you, @ESA_Rosetta! I did it! I
became the first spacecraft to land on a comet & study it! But it's not
over yet...
send @ESA_Rosetta #CometLanding You've done a great job Philae, something
no spacecraft has ever done before.
send  @Philae2014 #CometLanding I'm feeling a bit tired, did you get all
my data? I might take a nap...
list #MoonLanding
list #CometLanding
send @Wallace #MoonLanding Everything seems to be under contro-o-ol!
send @Wallace #MoonLanding Ooh... seconds to blast-off.
send @Wallace #MoonLanding No crackers, Gromit! We've forgotten the
crackers!
send @Wallace #MoonLanding Hold on, Gromit! Hold on!
send @Wallace #MoonLanding Adjust angle of thrust. Steady now. Easing up.
send @Wallace #MoonLanding Ah! Plate. Knife. Cracker.
send @Wallace #MoonLanding I don't know, lad. It's like no cheese I've
ever tasted. Let's try another spot.
list #Cheese
list #MoonLanding
quit
```

4.2. Dados de saída

```
NULL
@Armstrong #MoonLanding : That's one small step for man; one giant leap
for mankind.
@Armstrong #MoonLanding : Yes, the surface is fine and powdery. I can
kick it up loosely with my toe.
@Armstrong #MoonLanding : Ah... There seems to be no difficulty in moving
around - as we suspected.
NULL
@Philae2014 #CometLanding : Thank you, @ESA_Rosetta! I did it! I became
the first spacecraft to land on a comet & study it! But it's not over
yet...
@ESA_Rosetta #CometLanding : You've done a great job Philae, something no
spacecraft has ever done before.
@Philae2014 #CometLanding : I'm feeling a bit tired, did you get all my
data? I might take a nap...
@Wallace #Cheese : Gromit, that's it! Cheese! We'll go somewhere where
there's cheese!
@Wallace #Cheese : Everybody knows the moon's made of cheese.
@Wallace #MoonLanding : Everything seems to be under contro-o-ol!
@Wallace #MoonLanding : Ooh... seconds to blast-off.
@Wallace #MoonLanding : No crackers, Gromit! We've forgotten the
crackers!
@Wallace #MoonLanding : Hold on, Gromit! Hold on!
@Wallace #MoonLanding : Adjust angle of thrust. Steady now. Easing up.
@Wallace #Moonlanding : Ah! Plate. Knife. Cracker.
@Wallace #MoonLanding : I don't know, lad. It's like no cheese I've ever
tasted. Let's try another spot.
```

5. Exemplos e Material de Apoio

Na página da disciplina, secção “Material de apoio – Projectos” são fornecidos a título de exemplo ficheiros contendo diferentes sequências de input válidas, bem como os correspondentes ficheiros de saída produzidos. Estarão disponíveis ainda outros materiais de apoio à resolução do projecto.

6. Compilação e Execução do Programa

O compilador a utilizar é o **`gcc`** com as seguintes opções de compilação:

```
-ansi -Wall -pedantic
```

Para compilar o programa deve executar o seguinte comando:

```
$ gcc -ansi -Wall -pedantic -o proj2 *.c
```

o qual deve ter como resultado a geração do ficheiro executável **`proj2`**, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal. Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de aviso (warnings).
O programa deve ser executado da forma seguinte:

```
$ ./proj2 < test1.in > test1.myout
```

Posteriormente poderá comparar o seu output com o output previsto usando o comando **`diff`**:

```
$ diff test1.out test1.myout
```

7. Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.

- Efectue o upload de um ficheiro arquivo com extensão `.zip` que inclua os ficheiros fonte (`.c`) e, se for o seu caso, os cabeçalhos (`.h`) que constituem o programa. Para criar um ficheiro arquivo com a extensão .zip deve executar o seguinte comando na directoria onde se encontram os ficheiros com extensão .c, criados durante o desenvolvimento do projecto:
  ```
  $ zip proj2.zip *.c
  ```
  ou
  ```
  $ zip proj2.zip *.c *.h
  ```

- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.

- O sistema não permite submissões com menos de 10 minutos de intervalo para o mesmo grupo. Exemplos de casos de teste serão oportunamente fornecidos.

- Data limite de entrega do projecto: Segunda-feira, 1 de Dezembro de 2014 (23h59m). Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretende que seja avaliada.

8. Avaliação do Projecto

8.1. Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

- A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.

- A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, indentação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto. Nesta componente será também utilizado o sistema **`valgrind`** por forma a detectar fugas de memória ("memory leaks") ou outras incorrecções no código, que serão penalizadas. Aconselha-se por isso que os alunos utilizem este sistema para fazer debugging do código e corrigir eventuais incorrecções antes da submissão do projecto.

- Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

8.2. Atribuição Automática da Classificação

- A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados num computador com o sistema operativo GNU/Linux. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 64 Mb, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

- Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

- Não será disponibilizada informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.