BCC202 – Estruturas de Dados I (2023-02)

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (`www.decom.ufop.br/`)

AULA PRÁTICA 06

- **Data de entrega: Até 21 de novembro às 23:55.**

- Procedimento para a entrega:.

1. Submissão: via ***Moodle***.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via ***Moodle***.
6. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.
7. Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado.
8. Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.
9. A avaliação considerará o tempo de execução e o percentual de respostas corretas.
10. Eventualmente, serão realizadas entrevistas sobre os estudos dirigidos para complementar a avaliação.
11. Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada.
12. Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.
13. Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.
14. Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos.
15. Não serão considerados algoritmos parcialmente implementados.

- Bom trabalho!

Lista de amigos do Luiggy

Luiggy gosta de fazer amizades e sempre está expandindo sua lista de amigos na rede social *ListBook*. O *ListBook* permite que você inclua novos amigos em sua lista de amizade e os indique também para outros amigos da sua rede, tudo isso dinamicamente, sem limites de amigos na sua rede. Como Luiggy é seu amigo, ele pediu que você criasse um programa para facilitar a vida dele com esta tarefa. Para isso, Luiggy teve a seguinte ideia:

- O programa deverá ler a lista atual de amigos de Luiggy;
- O programa deverá ler uma nova lista de amigos de Luiggy;
- O programa deverá ler o nome de um amigo da lista atual que receberá a nova lista como indicação de amigos.

Considerações

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.

- Há vários casos de teste. Você terá acesso (entrada e saída) de casos específicos para realizar os seus testes localmente.

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Cada uma das listas de amigos deve ser armazenada em uma respectiva **lista encadeada** para cada caso de teste. A criação da lista de amigos final deve ser realizada com base na manipulação das duas listas encadeadas originais. As funções *TLista_Inicia*, *TLista_InsereFinal*, *TLista_RetiraPrimeiro* e *TLista_Imprime* são as mesmas vistas na aula teórica com modificações mínimas para tratar strings. As funções *TLista_Esvazia* (remove cada elemento de uma lista e libera a memória), *TLista_append* (acrescenta o conteúdo de uma lista ao final de outra, apenas manipulando ponteiros) e *TLista_include* (inclui o conteúdo de uma lista em outra, a partir de uma posição específica, apenas manipulando ponteiros) devem ser implementadas pelo aluno. A função *TLista_EhVazia* foi adaptada do material original.

Você deve ler em uma única linha a lista de amigos de Luiggy, contendo somente o primeiro nome e separados por um espaço em branco. Na segunda linha é informada a nova lista de amigos, novamente, contendo somente o primeiro nome e separados por um espaço em branco. Na terceira e última linha, é informado o nome do amigo para o qual a nova lista de amigos será indicada. Caso não se queira indicar para ninguém a nova lista de amigos, a última linha conterá a palavra "nao" (sem acento e sempre em letras minúsculas).

Seu programa deverá exibir a lista de amigos de Luiggy atualizada, ou seja, a união da lista original com a nova lista de amigos. Entretanto, se houver indicação para um amigo que já consta da lista original, os amigos da nova lista deverão ser inseridos **antes** do nome deste amigo. Caso não haja indicação, os novos nomes deverão ser inseridos no fim da lista de amigos original.

Entrada

`Jones Pedro Carlos Lucas`
`Juca Valdineia Jovander`
`Carlos`

Saída
`Jones Pedro Juca Valdineia Jovander Carlos Lucas`

Entrada

`Jones Pedro Carlos Lucas`
`Juca Valdineia Jovander`
`nao`

Saída
`Jones Pedro Carlos Lucas Juca Valdineia Jovander`

Diretivas de Compilação

```
$ gcc -c lista.c -Wall
$ gcc -c pratica.c -Wall
$ gcc lista.o pratica.o -o exe
```

Avaliação de *leaks* de memória

Uma forma de avaliar se não há *leaks* de memória é usando a ferramenta valgrind. Um exemplo de uso é:

```
gcc -g -o exe *.c -Wall; valgrind --leak-check=yes -s ./exe < casoteste.in
```

Espera-se uma saída com o fim semelhante a:

```
==38409== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Para instalar no Linux, basta usar: `sudo apt install valgrind`.