Universidade Federal da Bahia
MATD04 - Estruturas de Dados
Primeiro Trabalho Prático
Prof. Flávio Assis
Semestre 2023.2 - 1 de outubro de 2023

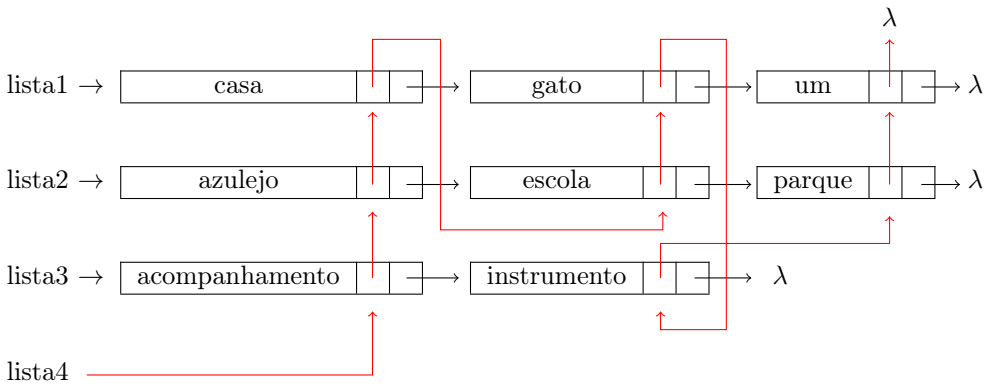
Listas de Palavras

1 Descrição Geral do Trabalho

Neste trabalho deverá ser implementado um programa que armazena palavras em listas simplesmente encadeadas e permite que consultas sobre essas palavras sejam feitas. O programa terá como entrada um conjunto de comandos a serem executados, como descrito na próxima seção. O programa deve ler da entrada padrão e escrever os resultados na saída padrão.

O programa manterá três listas básicas: uma, que armazenará palavras com até 5 letras; outra, que armazenará palavras de 6 a 10 letras; e uma terceira, que armazenará palavras com mais de 10 letras. Cada uma dessas listas deve ser implementada como uma *lista simplesmente encadeada* e deverá manter as palavras em ordem alfabética crescente (de A para Z). Adicionalmente, os nós dessas três listas devem estar encadeados adicionalmente em uma quarta lista (também simplesmente encadeada), que conecta os nós de todas as palavras em ordem alfabética crescente. Esta quarta lista ***não pode ser implementada de maneira separada das demais!***

A figura abaixo ilustra um exemplo. As três listas básicas são: *lista1* (referencia palavras com até 5 letras); *lista2* (palavras de 6 a 10 letras); e *lista3* (palavras com mais de 10 letras). A lista *lista4* conecta todas as palavras, seguindo a ordem alfabética (referências representadas em vermelho). Observe que cada nó terá dois campos (apontadores) para referenciar um outro nó: um para referenciar o próximo elemento de sua lista básica; e outro para referenciar o nó que contém a próxima palavra na *lista4*.



2 Formato da Entrada e Saída

A entrada constará de uma sequência de comandos. Nos comandos, uma palavra será formada apenas por letras minúsculas, sem acento e sem cedilha. Uma palavra não terá mais do que 29 caracteres.

O formato de cada comando está descrito a seguir.

1. **insere palavra:** este comando consistirá de duas linhas. A primeira conterá a letra 'i'. A segunda linha conterá uma palavra.

   Se a palavra já estiver em uma das listas, o programa deve gerar na saída: *'palavra ja existente:'*, seguida de um espaço, seguido da palavra digitada. Se não estiver, o programa deve inserir a palavra em uma das listas básicas, de acordo com o número de letras, e na lista que encadeia todas as palavras. Em seguida, o programa deve gerar na saída a sequência: *'palavra inserida:'*, seguida de um espaço, seguido da palavra.

2. **lista palavras em lista:** este comando consistirá de uma linha contendo a letra 'l', seguida por uma linha contendo um número. O comando listará as palavras de uma das listas, uma por linha, em ordem alfabética, de acordo com o número da segunda linha: se for 1, lista as palavras de 1 a 5 letras; se for 2, lista as palavras de 6 a 10 letras; se for 3, as palavras com mais de 10 letras; e 4, lista todas as palavras.

   Se não houver palavras que atendam ao comando, o programa deve gerar na saída a sequência: *'lista vazia'*.

3. **lista palavras por número de letras:** este comando consistirá de uma linha contendo a letra 'x', seguida por uma linha contendo um número (qualquer valor inteiro positivo). O comando lista as palavras armazenas nas listas que possuem exatamente o número de letras indicado na segunda linha do comando.

   Este comando apresenta na saída uma palavra por linha, em ordem alfabética. Caso não haja palavras com o número de letras dado, o programa gera na saída a sequência *'lista vazia'*.

4. **lista palavras em ordem alfabética:** este comando será formado por três linhas. Na primeira linha ocorrerá a letra 'o'. Na segunda e na terceira linha ocorrerá, em cada uma, uma letra. Sejam l_1 e l_2, respectivamente, as letras que ocorrem na segunda e na terceira linha. O programa deve retornar na saída as palavras que começam com as letras que vão de l_1 a l_2 no alfabeto (incluindo l_1 e l_2), em ordem alfabética, uma por linha. Caso não haja palavras a serem listadas, o programa deve gerar na saída a sequência: *'lista vazia'*.

   Considere que l_1 é igual a ou vem antes de l_2 no alfabeto.

5. **remove palavra:** este comando consistirá de uma linha contendo a letra 'r', seguida por uma segunda linha contendo uma palavra.

   Se não houver a palavra armazenada nas listas, o programa deve gerar na saída a sequência: *'palavra inexistente:'*, seguida de um espaço, seguido da palavra. Se houver, o nó contendo a palavra deve ser removido. Em seguida, o programa deve gerar na saída a sequência: *'palavra removida:'*, seguida de um espaço, seguido da palavra.

6. **término da seqüência de comandos:** a seqüência de comandos será terminada por uma linha com a letra 'e'.

3 Observações

- A implementação terá que, necessariamente, usar a estrutura de lista encadeada descrita acima.
- Os comandos devem gerar as saídas **somente** através do acesso às estruturas de listas encadeadas e seus (atributos), como descrito acima. Portanto, não poderão ser usadas listas de Python, nem métodos de ordenação existentes. **O uso desses mecanismos será considerado erro de programação (não atendimento à especificação) e, portanto, o trabalho será parcial ou totalmente desconsiderado.**
- Somente podem ser usados recursos da própria linguagem. Recursos adicionais somente podem ser utilizados se autorizados pelo professor. Em particular, nenhum recurso que não seja de biblioteca padrão da linguagem poderá ser usado.

- O programa não deve gerar nenhum caractere a mais na saída, além dos indicados acima. Em particular, **o programa não deve conter menus**.
- Não pode haver espaço entre linhas na saída. A saída deve apresentar os caracteres em letras minúsculas e sem acento.
- Trabalho individual.
- Linguagens de programação permitidas: C, C++, Java ou Python.
  - **Importante:** Para as linguagens C, C++ e Java, somente trabalhos feitos utilizando-se os seguintes compiladores serão aceitos:
    - C: gcc ou djgpp
    - C++: g++ ou djgpp
    - Java: compilador java do JDK (mais recente)

  No caso de Python, deve ser indicada a versão utilizada.

  Não serão compilados trabalhos em outros compiladores! Erros ocasionados por uso de diferentes compiladores serão considerados erros do trabalho!
- Data de entrega: **29/10/2023 - prazo inadiável! Somente serão aceitos trabalhos entregues até essa data, até 23:59h!**
- O(A) aluno(a) deverá submeter seu trabalho através do *moodle*.