==========

SSC0603 – ED1 – Prof. Fernando Osório – ICMC / USP 2020-2
Prof. Fernando Osório – Estrutura de Dados 1 (Eng.Comp.)
Run.Codes: Trabalho Prático - TP-03 Versão TP-03 v.01
Trabalho: Tradutor com Auto-Completar (AVL Trees)

==========

TP03 - Tradutor com Autocompletar em AVL Tree

>> Definição Geral do Trabalho: TADs Árvores Binárias Ordenada e Balanceada (AVL Tree)
(*) Programa implementado na Linguagem de programação "C"
(**) Podem ser usadas as bibliotecas (rotinas) do TAD do livro do Backes (vai precisar adaptar as rotinas originais), ou mesmo, rotinas de outros autores que implementem Árvores Binárias Ordenadas e Balanceadas criadas com Alocação Dinâmica (AVL ou outro tipo de Arv. Balanceada).

Este trabalho é relacionado ao uso de uma **lista não-linear encadeada dinâmica do tipo árvore binária ordenada e balanceada (AVL),** e também pode incluir o uso de **funções recursivas** para manipular/percorrer as árvores binárias, porém deve seguir a especificação fornecida abaixo.

ESPECIFICAÇÃO DO TRABALHO

ARQUIVO:
O programa irá ler os dados de um arquivo de nome "dict.txt", criando a estrutura de dados inicial (Árvore Binária Ordenada e Balanceada), depois serão lidos dados do teclado (stdin/scanf) com comandos de consulta e uso do dicionário, gerando a devida saída na tela (stdout/printf).

Note que vamos usar entrada de arquivo, com um arquivo que possui um nome específico "dict.txt" e que possui um formato exatamente como descrito mais abaixo. Além disso, teremos uma entrada pelo teclado e uma saída na tela, como já era feito usualmente nos programas que rodam no run.codes (interação com o programa).

O arquivo de entrada ("dict.txt") é um arquivo em formato texto com a seguinte estrutura, composta de "blocos" de 3 linhas que definem as palavras do dicionário:

1ª. Linha: Número de acesso que a palavra já teve (0 = nenhum, 1, 2, ... max: 99.999
Se o número for negativo, isso indica final de arquivo, ou seja, final da lista de palavras.

2ª Linha: Palavra origem (que será buscada para a tradução)

3ª. Linha: Palavra traduzida (que será retornada como resultado da tradução)

Observações: O número de acessos é um número entre 0 e 99.999

Se o número de acessos for -1, então isto indica final do arquivo de entrada.

As palavras têm no máximo 25 letras de comprimento, tanto original como traduzida

As palavras não possuem caracteres especiais, ou seja, sem acento, cedilha ou outra marca especial. Podem apenas possuir caracteres "normais" como: "_" (sublinhado), "+", "-", "$", "#", entre outros, porém não há espaços em branco entre letras. Os espaços em branco serão representados por "_" (sublinhado) se necessário.

Exemplo de Arquivo "dict.txt": (contendo apenas uma palavra com zero acessos)

```
0
book
livro
-1
```

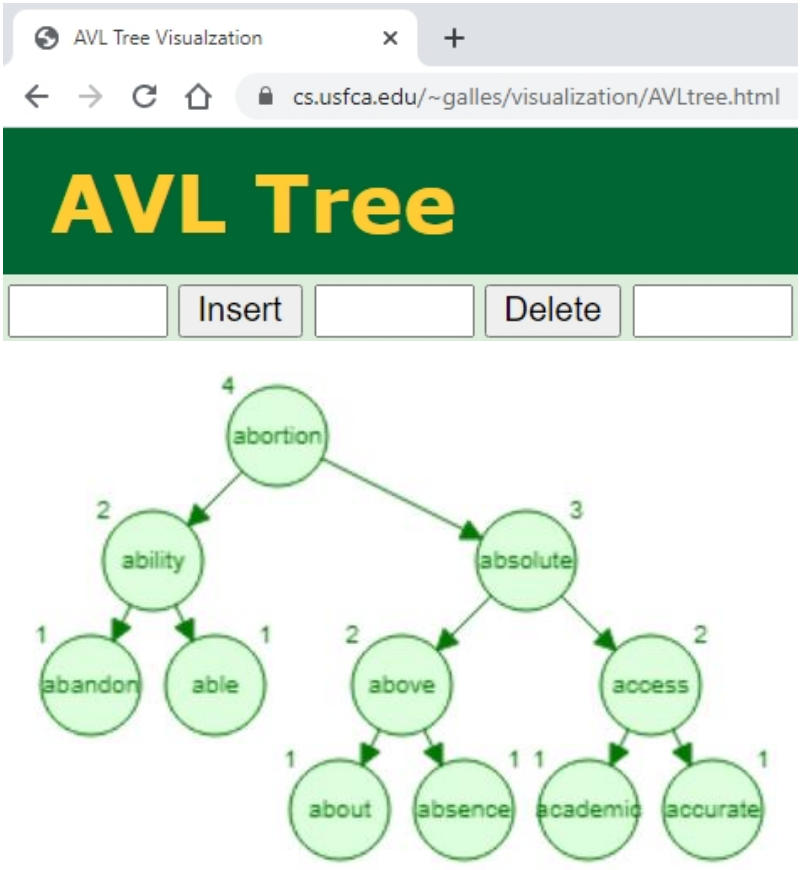
>> Exemplo de arquivo “dict.txt” de um dos casos de teste do trabalho:

> Todas as palavras estão com zero acessos, apresentam um para Inglês => Português e termina o arquivo de dicionário com -1. Note que os NODOS da árvore têm que armazenar os dados de <nro_de_acessos> <palavra_original> <palavra_traduzida>, criando uma árvore binária ordenada e balanceada (Sugere-se usar uma AVL)

```
0
the
o+a
0
book
livro
0
is
esta
0
on
sobre
0
table
mesa
-1
```

>> Exemplo de mais um arquivo “dict.txt” de um dos casos de teste do trabalho:

```
1
abandon
abandono
1
ability
habilidade
1
able
capaz
2
abortion
aborto
2
about
sobre
2
above
acima
5
absence
ausencia
4
absolute
absoluto
2
academic
academico
2
access
acesso
2
accurate
preciso
-1
```



https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

> Algumas palavras deste arquivo já foram consultadas em traduções realizadas anteriormente, e estão com seu contador de acessos maior que zero.

CONSULTAS:

As consultas aos dados da árvore e do dicionário serão realizadas por entradas usando a entrada padrão do run.codes, ou seja, como se alguém estivesse digitando dados no teclado (stdin/scanf).

O usuário deve digitar primeiramente um valor numérico (valor inteiro), representando o tipo de operação que deseja realizar:

0 (zero) – **Função de Tradução** de uma palavra. Neste caso, na linha seguinte de entrada o usuário digitará a palavra a ser buscada, traduzida e exibida na tela. NÃO esqueça de incrementar (somar +1) ao contador de acesso. Cada vez que uma palavra for usada na tradução, devemos somar 1 no contador de acessos. Por exemplo:

- Entrada: (o usuário digita um nro. seguido de enter, e uma palavra seguida de enter)

```
0
book
```

- Exibe na tela: (pula para próxima linha após cada palavra exibida => "\n")

```
livro
```

1 (um) até 99.999 **– Função de Auto-Completar** as palavras buscadas, neste caso, na linha seguinte a entrada do usuário é a parte de uma palavra, com pelo menos 2 caracteres, onde devem ser buscadas e exibidas TODAS as ocorrências de palavras que tenham esta "parte" da palavra, MAS ATENÇÂO, só devem ser exibidas as palavras com pelo menos o número de acessos que foi lido (valor inteiro entre 1 e 99.999), ou seja, se o valor for 3, só devem ser exibidas palavras que possuem pelo menos 3 acessos ou mais em seu contador. Por exemplo:

- Entrada:

```
2
acc
```

- Exibe na tela: (autocompletar de "acc" com pelo menos 2 acessos)

```
access
accurate
```

-1 (menos 1): **Função de Término de consultas.**
Indica o término da sequência de entradas de dados do usuário.

Exemplo de uma consulta mais completa usando o primeiro dicionário ("the book is on the table"):

Entrada:

```
0
table
0
book
-1
```

Saída:

```
mesa
livro
```

Entrada: Note que "book" foi acessado UMA VEZ no dicionário!

```
0
table
0
book
1
bo
-1
```

Saída:

mesa
livro
book

> Neste caso, “mesa” e “livro” são as traduções solicitadas (função “zero”) e “book” é o resultado da consulta de Auto-Completar a palavra buscada “bo” que tenha pelo menos 1 acesso anterior, resultando na palavra “bo”.

Se considerarmos o dicionário mais completo (de 11 palavras começando com a letra “a”), podemos ter o seguinte exemplo de consulta do dicionário:

Entrada: (realiza apenas consultas de tradução)

0
accurate
0
access
0
about
0
academic
0
ability
-1

Saída:

preciso
acesso
sobre
academico
habilidade

> Mais um exemplo com este mesmo dicionário:

Entrada: (realiza apenas o uso da função de auto-completar)

2
abo
3
ab
2
acc
-1

Saída:

abortion
about
above
absence
absolute
access
accurate

> abortion, about, above vem de “abo”; absence e absolute de “ab”; e access e accurate de “acc”
> Note que o autocompletar informa as palavras em ordem alfabética: foi pensado considerando a rotina de percorrer a árvore “emOrdem”. Isso importa na hora da correção automática (run.codes)

IMPORTANTE:

Você já deve ter percebido que a função de busca por strings terá que ser específica e implementada por você, ou seja, você vai ter que criar uma "espécie" de STRCMP, para comparar as letras iniciais da string até onde for possível, no caso do auto-completar.

A função de busca de palavras no auto-completar deve OBRIGATORIAMENTE deixar de visitar através da recursão palavras (e nodos) que já não tenham mais chance de se encaixar no padrão dado, porque estão abaixo ou acima na ordenação e fora do escopo pesquisado. Deve ser realizado um CORTE na recursão destes nodos que estão fora do CONJUNTO de palavras buscadas (Redução do Espaço de Busca).

ENTREGA DO TRABALHO:

>> O trabalho deve ser entregue via RUN.CODES

>> A definição do trabalho e exemplos de casos de teste estão disponíveis na página da Wiki:

http://wiki.icmc.usp.br/index.php/TrabPrat_SSC0603_2020(fosorio)

Data para entrega, dentro do prazo usual, ou seja, até o final do semestre: **21/12/2020**

```
===========
  F.Osório
  Nov.2020
===========
```