TRABALHO FINAL
Contando Calorias

1 Objetivo

O objetivo do trabalho é comparar o desempenho das Árvores Binárias de Pesquisa em uma aplicação de contagem de calorias.

2 Especificação da Aplicação

Uma nutricionista acompanha diversos pacientes em dieta. A fim de ajudá-los a controlar sua alimentação, ela pede que eles registrem em um arquivo todos os alimentos que ingerem diariamente. Ao final da semana, os pacientes enviam todos os arquivos para ela que precisa então calcular o total de calorias ingeridas por eles dia a dia. Para esse cálculo, ela utiliza uma tabela que possui os nomes dos alimentos e a quantidade de calorias que eles possuem (em uma porção de 100 gramas). Este é um processo trabalhoso e sujeito a erros. Por isso, ela decidiu automatizá-lo.

- Sua tarefa é projetar uma aplicação que recebe um arquivo texto contendo os alimentos ingeridos pelo paciente e uma tabela de alimentos e suas calorias. A tabela de calorias deverá ficar armazenada em uma árvore. Você deve criar **diferentes versões da aplicação** utilizando **pelo menos duas** das quatro árvores vistas em aula (ABP, AVL, Rubro-Negra ou Splay).
- A tabela de calorias deve ser carregada na árvore na ordem em que as palavras estão no arquivo.
- A aplicação **não é case sensitive,** letras maiúsculas e minúsculas devem ser consideradas iguais.
- A lista de alimentos ingeridos pelos pacientes **não deve** ser armazenada em nenhuma estrutura. Cada alimento será usado apenas para consultar a árvore e encontrar suas respectivas calorias.
- Seu programa deverá ser chamado a partir da linha de comando (passando parâmetros para o main).

Exemplo de chamada:

```
C:\contador_calorias 1000Shuffled.csv day1.csv saída_day1.txt
```

As entradas e saídas da sua aplicação são:

Entradas:

(i) nome do arquivo com a tabela de calorias e
(ii) nome arquivo com os alimentos ingeridos.

Saídas:

(i) arquivo de saída com o número de calorias ingeridas e

(ii) estatísticas sobre o processo de geração da árvore e realização de consultas (número de nodos, altura da árvore, número de rotações e número de comparações realizadas durante as consultas).

 Os arquivos de entrada são arquivos de texto com a extensão csv e tem um alimento por linha. O arquivo com a tabela de calorias tem o alimento e seu número de calorias em uma porção de 100g. O arquivo com os alimentos ingeridos tem o nome alimento e a quantidade ingerida em gramas.

Por exemplo:

Tabela de calorias (alimento;calorias_por_100g) Arquivo: 1000shuffled.csv	Alimentos ingeridos (alimento;qtde_em_gramas) Arquivo: day1.csv
Plum Juice;71 Pearl Barley;352 Bran Muffins;270 Barq's;46 Mozzarella Pizza;249 Rack of Pork;241 Rosemary;131 Brownies;405 Potato Fritter;185 Turkey Breast;104 Oatmeal Raisin Cookies;435 Muesli;336 Cherries;50 Seafood Pizza;245 Cornbread;179 …	Activia;150 Tangerine;110 Bagel;120 Coffee;200 Beef Fillet;200 Rigatoni;150 Banoffee Pie;100 Arugula;50 Caesar Dressing;5 Camembert;30 Chocolate;50 White Wine;200

Arquivo de saída
Saída_day1.txt

```
Calorias calculadas para day1.csv usando a tabela 1000shuffled.csv.

150g de activia (74 calorias por 100g) = 111 calorias
110g de tangerine (53 calorias por 100g) = 58 calorias
120g de bagel (257 calorias por 100g) = 308 calorias
200g de coffee (1 calorias por 100g) = 2 calorias
200g de beef fillet (189 calorias por 100g) = 378 calorias
150g de rigatoni (353 calorias por 100g) = 529 calorias
100g de banoffee pie (395 calorias por 100g) = 395 calorias
50g de arugula (25 calorias por 100g) = 12 calorias
5g de caesar dressing (429 calorias por 100g) = 21 calorias
30g de camembert (300 calorias por 100g) = 90 calorias
50g de chocolate (529 calorias por 100g) = 264 calorias
200g de white wine (82 calorias por 100g) = 164 calorias

Total de 2332 calorias consumidas no dia.

========  ESTATÍSTICAS ABP ============
Numero de Nodos: 1000
Altura: 22
Rotações: 0
Comparações: 310
```

```
========  ESTATÍSTICAS AVL ============
Numero de Nodos: 1000
Altura 12
Rotações: 449
Comparações: 248
```

 O número de **comparações** realizadas com os elementos da árvore **deve** ser calculado pela função a seguir (onde comp é uma variável global que acumula o número de comparações).

```
pNodoA* consulta(pNodoA *a, char *chave){
    while (a!=NULL){
         comp++;
        if (!strcmp(a->alimento,chave)){
            comp++;
            return a;
        }
        else {
           comp++;
           if (strcmp(a->alimento,chave)>0)
              a = a->esq;
            else
              a = a->dir;
        }
    }
    return NULL;
}
```

 Arquivos de teste (entradas e a respectiva saída) estão disponíveis no Moodle. No dia da apresentação, novos conjuntos de arquivos serão fornecidos.

2 Requisitos

- É necessário elaborar um relatório **detalhado** com a análise comparativa do desempenho das duas árvores. Utilize recursos como **tabelas** e **gráficos** para dar suporte às suas conclusões.
- É recomendável testar com **diferentes tamanhos de arquivos e formas de ordenação** para permitir uma melhor análise do comportamento das árvores.
- O trabalho deve ser feito, preferencialmente, em duplas. Também aceitaremos trabalhos feitos individualmente e de duplas cujos integrantes sejam de turmas diferentes.
- A linguagem de programação aceita é C (Não é C++ nem C#).

3. Entrega e Apresentação

- **06 de outubro de 2022 [entrega antecipada]** apresentação em aula presencial e entrega pelo Moodle. *Aqueles que apresentarem nesta data, a nota do trabalho será calcular nota do trabalho + 10% da nota tirada no trabalho (mesmo que ultrapasse 10,0)*
- **11 de outubro de 2022** apresentação em aula presencial e entrega pelo Moodle. *Nota máxima 10,0*

4. Critérios de Avaliação

O trabalho deve ser realizado em duplas e deverá ser apresentado e defendido na data prevista.

Para a avaliação serão adotados diversos critérios:

- funcionamento (Peso: 30%);
- organização e documentação do código (Peso: 30%); e
- relatório (Peso: 40%).

5. Dicas

- Há um exemplo de código no Moodle (com um vídeo explicativo) que mostra como fazer a passagem de parâmetros para a função main.
- Para ler cada linha dos arquivos de entrada, utilize a função `fgets`.
- Para tokenizar as linhas (separar o nome do alimento do seu número de calorias ou quantidade), utilize a função `strtok`. O separador utilizado no arquivo csv é ";".
- A ordem lexicográfica dos nomes dos alimentos determinará a organização da árvore, *i.e.*, a ordem em que aparecem no dicionário. Por exemplo, se a palavra *"morango"* for a raiz da árvore, então a palavra *"alface"* deve aparecer na subárvore esquerda da raiz (utilize a função `strcmp`) para comparar as strings.

Importante:

Este trabalho deverá representar a solução da dupla para o problema proposto. O plágio é terminantemente proibido e a sua detecção irá zerar a nota do trabalho. É permitido reusar código disponibilizado em aula ou até mesmo encontrado na Internet desde que todas as fontes sejam referenciadas no código e no relatório.