UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
DIM0806 - ESTRUTURAS DE DADOS E ALGORITMOS
PROF. BRUNO MOTTA DE CARVALHO

Projeto 1 - Árvores Rubro-Negras - Trabalho Individual
Entrega: 19/11/2018 Horário: até às 23:59 Local: SIGAA

A Árvore Rubro-Negra é um tipo de árvore binária de busca balanceada bastante utilizada. Ela é uma estrutura que se ajusta a medida em que novos elementos são inseridos, de modo a manter um balanceamento entre diferentes sub-árvores. Vocês devem enviar seu código e um relatório em um arquivo comprimido pelo SIGAA.

Vocês deverão **implementar em C/C++** uma árvore rubro-negra que será utilizada para buscar palavras de um dicionário. Você pode assumir que o tamanho máximo das palavras a serem armazenadas é de 20 caracteres. Seu programa deverá ler um arquivo texto cujo nome será especificado na linha de comando, montar a árvore rubro-negra correspondente e iniciar um laço que lê palavras a serem buscadas na árvore. Um exemplo da chamada do programa pode ser visto abaixo.

```
> RBTree dicionario.txt
```

Cada linha do arquivo texto contém uma palavra com até 20 dígitos e um número, que pode ser 0 ou 1. Quando esse número for igual a 1, vocês devem inserir a palavra na árvore, caso ela não esteja presente. Se a palavra já estiver na árvore, seu programa deve imprimir uma mensagem e não inserir uma nova cópia da palavra. Se o número for igual a 0, vocês devem remover a palavra da árvore, caso ela já tenha sido incluída na árvore e não removida. Se a palavra não existir na árvore, uma mensagem deve ser imprimida relatando o fato. *Após uma deleção, as funções RBPrint e RBCheck devem ser chamadas para a verificação da estrutura da árvore.*

A entrada de um string vazio na busca resulta na saída do laço e na chamada às funções RBPrint e RBCheck. Você deverá implementar as funções elencadas abaixo, além das funções auxiliares necessárias. Vocês devem se basear no código disponibilizado no livro do Cormen.

- `void RBInsert(RBTree T, RBElement z)` - Insere o elemento `z` na árvore `T` mantendo as propriedades de uma Árvore Rubro-Negra. Utiliza a função auxiliar `void RB-Insert-Fixup(RBTree T, RBElement x)`, que por sua vez utiliza as funções `void Right-Rotate(RBTree T, RBElement z)` e `void Left-Rotate(RBTree T, RBElement z)`.

- `void RB-Delete(RBTree T, RBElement z)` - Deleta o elemento `z` da árvore `T` mantendo as propriedades de uma Árvore Rubro-Negra. Utiliza as seguintes funções auxiliares: `void RB-Transplant(RBTree T, RBElement z, RBElement y)`, `RBElement Tree-Minimum(RBTree T)` e `RB-Delete-Fixup(RBTree T, RBElement x)`, que por sua vez utiliza as funções `void Right-Rotate(RBTree T, RBElement z)` e `void Left-Rotate(RBTree T, RBElement z)`.

- `RBElement Search(RBTree T, String c)` - Busca pelo string `c` na árvore `T`, imprime o resultado da busca e retorna um ponteiro para o elemento cuja chave é `c`.

- `void RBPrint(RBTree T)` - Imprime a lista ordenada de todas as chaves armazenadas na árvore `T`.

  `...cavalo doido drone esperança facão fortuito ...malária ...`

- `void RBCheck(RBTree T)` - Imprime, para cada nó da árvore, as seguintes informações: chave do nó pai, chave própria, cor do nó, altura negra, chaves dos filhos. A árvore deve ser visitada em pré-ordem. Você deve imprimir um nó por linha, como pode ser visto no exemplo abaixo:

  `(NIL, fortuito, preto, 3, esperança, malária)`

  `(fortuito, esperança, vermelho, 3, doido, facão)`

  `(esperança, doido, preto, 2, cavalo, drone)`

  ⋮ ⋮ ⋮ ⋮ ⋮ ⋮