Resolva cada exercício em folhas separadas

1. Considere uma lista ligada de palavras que forma uma frase sem pontuação. Considere ainda que tem acesso a um dicionário de palavras. O objetivo é desenvolver um método que receba a lista e o dicionário e devolva a lista alterada com os erros assinalados. No entanto, há um detalhe, dois **erros iguais** devem ser **identificados** da **mesma forma**. Por exemplo, se entrar no método uma lista ligada com os seguintes elementos:

`{os,campos,são,berdes,os,zolhos,dela,são,berdes,e,os,meus,zolhos,tamem}`

A lista ligada de saída deverá ter os seguintes elementos:

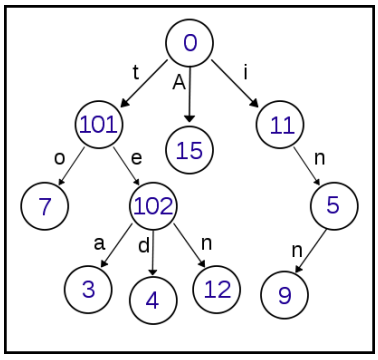
`{os,campos,são,`**`ERRO_01`**`,os,`**`ERRO_02`**`,dela,sao,`**`ERRO_01`**`,e,os,meus,`**`ERRO_02`**`,`**`ERRO_03`**`}`

Crie o código do método:

```
DoublyLinkedList<String> checkErrors(DoublyLinkedList<String> str, Set<String> dictionary)
```

de forma a que este devolva a lista com as palavras erradas assinaladas.
Nota: Será valorizada uma resolução com complexidade linear.

2. Uma trie é uma estrutura de dados muito usada em *Information Retrieval* para reconhecimento de sequências de carateres. Genericamente uma trie pode ser vista como uma árvore n-ária com símbolos nos ramos e valores numéricos nós, sendo que neste caso, os **nós terminais** têm valores numéricos **menores que 100**, os **nós não terminais** têm valores numéricos **maiores que 100** e o **nó inicial** tem o **valor zero**.



Para reconhecer uma sequência de carateres a *trie* começa no nó inicial e usa cada um dos carateres da sequência para transitar através do ramo com esse carater, caso o ramo exista. Se, no final da sequência estiver num nó terminal, a sequência é reconhecida. A *trie* acima representada reconhece as sequências: { to, tea, ted, ten, A, i, in, inn }. Já por exemplo as sequências { t, te, x, tent, Al } não são reconhecidas.

Resolva cada exercício em folhas separadas

Admita que a **trie** acima representada está guardada num **grafo orientado**, representação **map de adjancência**, e escreva um método que valide se uma sequência de carateres é reconhecida pela trie. Deve devolver o número correspondente ao nó terminal se a sequência for reconhecida ou -1 se não for.

```
public Integer checkSequence(Graph<Integer,Character> trie, Character[] sequence)
```

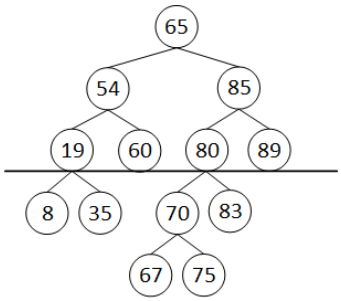
3. Considere o seguinte método implementado na **classe BST**:

```
protected E misterio (Node<E> node){
   if (node == null)
       return null;
   Node<E> nodeAux;
   for (nodeAux = node; nodeAux.getLeft()!=null; nodeAux = nodeAux.getLeft());
   return nodeAux.getElement();
}
```

a) Explique o que faz o método acima apresentado.

b) Analise o método quanto à sua complexidade temporal. Justifique.

4. Adicione à classe TREE<E> um método que devolva uma lista com os elementos da metade superior da árvore ordenados de forma descendente. Para a árvore abaixo o método deve devolver a lista {89, 85, 80, 65, 60, 54, 19}.



5. Escreva um método que verifica se uma sequencia de N inteiros obedece ao critério de ordenação de uma heap.