Problema 8

Criptografia de Dados

Algoritmos e Estruturas de Dados I

Tempo limite: 1s

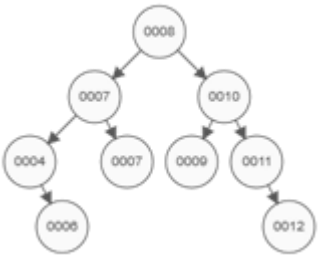
Dois estudantes desenvolveram um método inusitado de criptografar as palavras de suas mensagens. Eles resolveram codificar uma palavra em uma Árvore Binária de Busca (ABB) usando uma sequência de letras. Cada nó dessa árvore armazena uma letra e a quantidade de vezes (frequência) que ela ocorre em um *string*. Os **nós são posicionados na ABB com base na frequência de cada letra.**

Para dificultar a decodificação por alguém desconhecido, eles acrescentaram duas letras distintas e que não fazem parte da palavra original, na sequência das letras. Cada uma delas também possuem sua própria frequência.

A palavra decodificada é a sequência de percorrimento **em-ordem** (*in-order*) da ABB final (após as remoções das duas letras indicadas). Lembre-se que, conforme a definição, uma ABB tem as seguintes propriedades:

- Todos os elementos da subárvore esquerda têm valor menor que X;
- Todos os elementos da subárvore direita têm valor maior ou igual à X.

Por exemplo:



Deve-se escrever um programa que construa uma árvore binária de busca com base na frequência com que as letras aparecem na *string*. Logo após, deve-se exibir a palavra decodificada.

Entrada

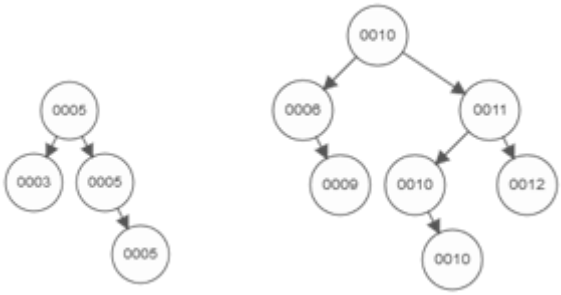
A entrada para cada teste se inicia informando as duas letras distintas que foram acrescentadas na *string* de entrada e que deverão ser removidas. Na segunda linha é fornecida a *string* de entrada de tamanho **N** ($1 \leq \boldsymbol{N} \leq 1024$). Cada letra tem frequência maior ou igual a 1 na *string*, ou seja, cada letra aparece no mínimo uma vez.

Saída

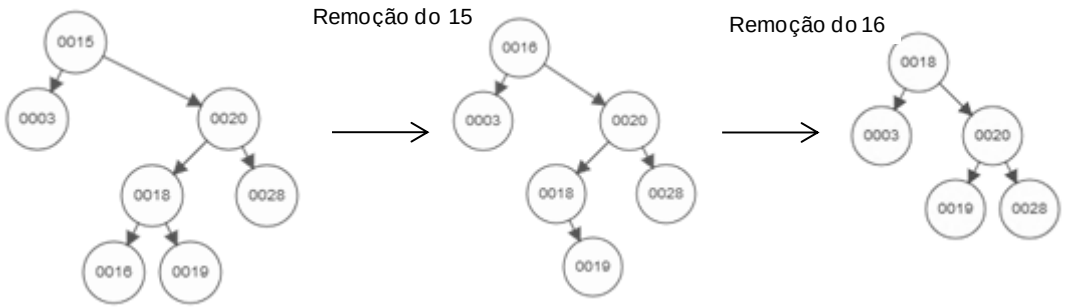
A saída do programa é exibida em duas linhas que correspondem, na primeira linha, a palavra formada pelo percorrimento em ordem da árvore binária de busca com todas as letras inseridas, ainda sem a remoção das duas letras indicadas. A segunda linha deve exibir a palavra decodificada (já com as remoções).

Dicas:

- A árvore proposta deve aceitar elementos repetidos. Nesse caso os elementos iguais serão alocados na subárvore direita do elemento encontrado. Por exemplo:



- No caso da remoção com dois filhos não nulos, deve-se seguir a regra de escolha do elemento mais à esquerda da subárvore direita (**sucessor**). Por exemplo:

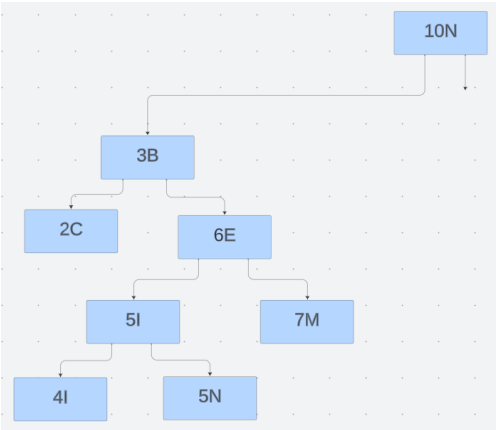


Restrições:

- O programa deve ser e escrito em C;
- Deve-se ter no código as funções de: inicialização, inserção, percorrimento em-ordem e remoção em árvores binárias de busca;
- Deve-se utilizar uma TAD Árvore Binária.
- Nesse problema serão consideradas apenas letras maiúsculas de A até Z;

- Para simplificação não haverá palavras a serem decodificadas (*string* de saída) com letras repetidas, como por exemplo: TRABALHO (tem duas letras A), ARARA (tem duas letras R e três letras A), entre outras.

- A sequência de inserção dos nós da Árvore Binária de Busca **DEVE** ser feita seguindo a **ordem alfabética das letras**, como por exemplo, no caso da string de entrada:
  MNMABINIEMMBBAJJEJJNCNAIMMIMEAAINEEAAEAACA
  Porém, o posicionamento de cada par, frequência-letra, na árvore binária deve ter como chave a frequência, conforme já explicitado.
  Para o caso do exemplo acima, teríamos os seguintes pares de letra-frequência ordenados alfabeticamente:
  10A, 3B, 2C, 6E, 5I, 4J, 7M, 5N
  A inserção dos dados na ABB deve seguir exatamente a sequência acima. Entretanto, a ABB deve posicionar seus nós de acordo com a frequência de cada letra. Dessa forma a árvore formada seria:



Exemplo de Entrada	**Exemplo de Saída**
D K EEEEELDDKDLATDLLALAAAADDKLTADAT	KTELDA TELA

Exemplo de Entrada	**Exemplo de Saída**
B J MNMABINIEMMBBAJJEJJNCNAIMMIMEAAINEEAAEAACA	CBJINEMA CINEMA

Exemplo de Entrada	**Exemplo de Saída**
C W YCACNRCYWNYCBRYRRRYCCCYNIAARRRYRYYCRCARYCCYCCCYCCI	BWINARYC BINARY