



Programação 2

2º Semestre

Trabalho Prático P4A

Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Um método de codificação de caracteres que consegue minimizar o número de bits usados é o denominado método de Huffman. Neste método, é criada uma árvore binária em que as folhas contêm os caracteres a codificar e o caminho desde o nó até uma folha é a codificação desse carácter (percorrer um ramo esquerdo corresponde ao bit 0 e um ramo direito ao bit 1).

Os nós das árvores são objectos da classe **FreqCrts**.

```
class FreqCrts {
    char character; // não é usado para nós que não são folhas
    int frequencia;
public:
    FreqCrts(char c='~', int freq=1) { character=c; frequencia=freq; }
    char getChar() { return character; }
    int getFrequencia() { return frequencia; }
    void incFrequencia() { frequencia++; }
    bool operator < (FreqCrts & fc2) { return frequencia < fc2.frequencia; }
    friend ostream &operator << (ostream &os, FreqCrts &fc1);
};

ostream &operator << (ostream &os, FreqCrts &fc1)
{
    os << fc1.character << " : frequencia=" << fc1.frequencia << endl;
    return os;
}
```

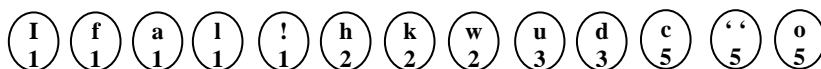
A árvore é criada da seguinte forma:

- i. Colocar os caracteres da frase em uma lista ordenada por frequência do carácter na frase. Os elementos da lista são árvores binárias com um único nó. Os nós das árvores são objectos da classe **FreqCrts**.
- ii. Percorrer a lista emparelhando os elementos desta dois a dois em uma árvore binária. Isto é:
 - o O primeiro e o segundo elemento da lista (X1 e X2) são retirados e dão origem a uma árvore com estas duas subárvores X1 e X2. A raiz da nova árvore X possui frequência igual à soma das frequências de X1 e X2.
 - o Colocar a nova árvore X ordenadamente na lista, isto é, após todas as árvores com raiz menor (valor de frequência inferior).
 - o Repetir até a lista possuir uma única árvore.

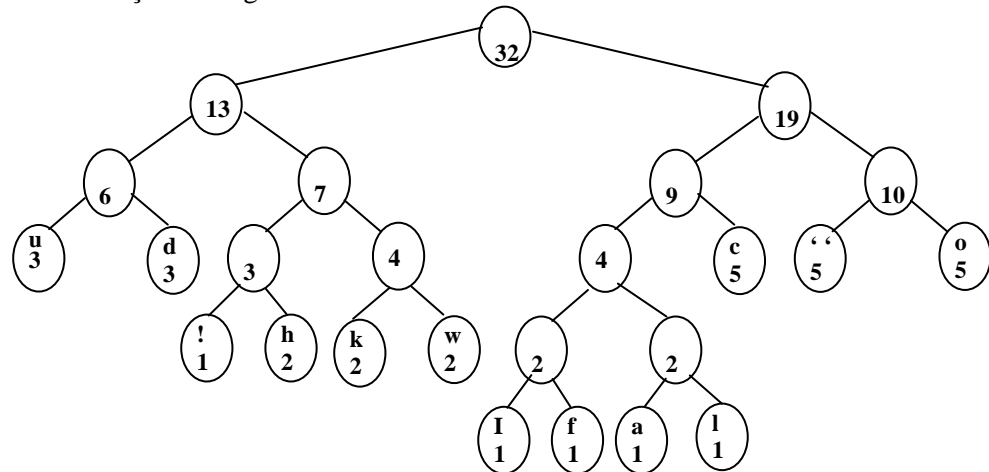
Como exemplo, considere a seguinte frase:

“If a woodchuck could chuck wood!”

A lista inicial definida no passo i) será a seguinte:



A árvore final de codificação é a seguinte:



a) Implemente a função que coloca uma árvore de forma ordenada na lista

```
void colocaLista(list<BinaryTree<FreqCrts> > &listArv,
                BinaryTree<FreqCrts> &novaArv)
```

Esta função coloca a árvore *novaArv* na lista *listArv* de forma ordenada. A árvore *novaArv* deve ser inserida após todas as árvores cuja raiz é inferior à de *NovaArv*, e antes de qualquer árvore de raiz igual ou superior à de *novaArv*.

b) Implemente a função que dada uma frase constrói a lista referida em i)

```
void criaLista(list<BinaryTree<FreqCrts> > &listArv, string &frase)
```

Esta função cria a lista *listArv*. Esta lista contém árvores binárias de um único nó que estão colocadas na lista de forma ordenada. O nó raiz (e único nó) de cada árvore contida na lista é um objecto da classe **FreqCrts** correspondente a um carácter diferente da frase a codificar (*frase*).

c) Implemente a função que cria a árvore final de codificação

```
void codifica(list<BinaryTree<FreqCrts> > &listArv)
```

Esta função emparelha os elementos/árvores da lista *listArv*, conforme descrito em ii). No final deste processo, a lista *listArv* possui uma única árvore que é a codificação final da frase.