```
#include "HuffmanNode.h"
using namespace std;

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//The Default Constructor
HuffmanNode::HuffmanNode(){
    letter = ' ';
    frequency = 0;
    left = right = NULL;
}


//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Costructor for Leaf nodes
HuffmanNode::HuffmanNode(char c, int i){
    letter = c;
    frequency = i;
    left = right = NULL;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//For constructing root node of two nodes .i.e Internal Nodes
HuffmanNode::HuffmanNode(HuffmanNode* left, HuffmanNode* right){
    this->left = left;
    this->right = right;
    frequency = left->getFrequency() + right->getFrequency();
    letter = NULL;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Destrctor
HuffmanNode::~HuffmanNode(){
    delete left;
    delete right;
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Copy Constructor
HuffmanNode::HuffmanNode(const HuffmanNode& rhs){
    //cout << "in copy constructor copying: " << rhs.letter << endl;

    if(rhs.left != NULL){
        left = new HuffmanNode();
        *left = *(rhs.left);
    }
    if(rhs.right != NULL){
        right = new HuffmanNode();
        *right = *(rhs.right);
    }
    code = rhs.code;
    letter = rhs.letter;
    frequency = rhs.frequency;
}


//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Prints the character followed a tab, followed the
ostream& operator<<(ostream& os, const HuffmanNode& rhs){
    os <<  "\'" << rhs.letter << " " << rhs.frequency << "\' ";
    return os;
}


//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Assignment operator
HuffmanNode& HuffmanNode::operator=(const HuffmanNode& rhs){
```

```
    if(!(this == &rhs)){            // Check for self-assignment

        code = rhs.code;
        letter = rhs.letter;
        frequency = rhs.frequency;

        delete left;
        delete right;

        if(rhs.left != NULL){
            left = new HuffmanNode();
            *left = *(rhs.left);
            if(rhs.right != NULL){
                right = new HuffmanNode();
                *right = *(rhs.right);
            }
        }
        else left = right = NULL;
    }

    return *this;
}
```