```
template <typename T>
void BinarySearchTree<T>::searchD(const T& item, bool& found,
                    Node<T>*& locPtr, Node<T>*& parent) const
{
    locPtr = root;
    parent = nullptr;
    found = false;
    while (!found && locPtr != nullptr) {
        if (item < locPtr->value) {
            parent = locPtr;
            locPtr = locPtr->left;
        }
        else if (locPtr->value < item) {
            parent = locPtr;
            locPtr = locPtr->right;
        }
        else {
            found = true;
        }
    }
}
```

```
template <typename T>
void BinarySearchTree<T>::deleteNode(const T& item)
{
	using namespace std;
	bool found;
	Node<T>* current;
	Node<T>* parent;
	searchD(item, found, current, parent);
	if (!found) {
		cout << "Item not in tree" << endl;
	}
	else {
		if ((current->left != nullptr) && (current->right != nullptr)) {
			Node<T>* currentSucc = current->right;
			parent = current;
			while (currentSucc->left != nullptr) {
				parent = currentSucc;
				currentSucc = currentSucc->left;
			}
			current->value = currentSucc->value;
			current = currentSucc;
		}
```

```
            Node<T>* subtree = current->left;
            if (subtree == nullptr) {
                subtree = current->right;
            }
             if (parent == nullptr)  {
                root = subtree;
            }
            else if (parent->left == current) {
                parent->left = subtree;
            }
            else  {
                parent->right = subtree;
            }
            delete current;
        }
}
```