## Section B

Given a BST and a target value, write a function to find a path from the root to the node with the given value. If the value does not exist in the BST, return an empty path.

**Task**: Implement a function `findPathToNode(root, value)` that takes a BST and a target value, returning a list of node values representing the path from the root to the node with the given value.

**Example**: Consider the following BST:

```
    10
    /  \
   5   15
  / \  / \
 1  7 12 18
```

Given the target sum 22, the expected output would be:

- Path 1: 5 -> 4 -> 11 -> 2
- Path 2: 5 -> 8 -> 4 -> 5

```
template<typename T>
bool findPathHelper(typename BST<T>::Node* root, T target, std::vector<T>&
path) {
    if (root == nullptr) {
        return false;
    }

    path.push_back(root->value);

    if (root->value == target) {
        return true;
    }

    if ((root->left != nullptr && findPathHelper(root->left, target, path))
||
        (root->right != nullptr && findPathHelper(root->right, target,
path))) {
        return true;
    }
```

```cpp
    path.pop_back();
    return false;
}

template<typename T>
std::vector<T> findPathToNode(typename BST<T>::Node* root, T target) {
    std::vector<T> path;
    findPathHelper(root, target, path);
    return path;
}
```