```cpp

// COS30008, Final Exam, 2022

#pragma once

#include "TernaryTree.h"

#include <stack>

template<typename T>
class TernaryTreePrefixIterator
{
private:
    using TTree = TernaryTree<T>;
    using TTreeNode = TTree*;
    using TTreeStack = std::stack<const TTree*>;

    const TTree* fTTree;                // ternary tree
    TTreeStack fStack;                  // traversal stack

public:

    using Iterator = TernaryTreePrefixIterator<T>;

    Iterator operator++(int)
    {
        Iterator old = *this;

        ++(*this);

        return old;
    }

    bool operator!=( const Iterator& aOtherIter ) const
    {
        return !(*this == aOtherIter);
    }

/////////////////////////////////////////////////////////////////////////
// Problem 4: TernaryTree Prefix Iterator

private:

    // push subtree of aNode [30]
    void push_subtrees( const TTree* aNode )
    {
        if (!(*aNode).getRight().empty())
        {
            fStack.push(const_cast<TTreeNode>(&(*aNode).getRight()));
        }
        if (!(*aNode).getMiddle().empty())
        {
            fStack.push(const_cast<TTreeNode>(&(*aNode).getMiddle()));
        }
        if (!(*aNode).getLeft().empty())
        {
            fStack.push(const_cast<TTreeNode>(&(*aNode).getLeft())); 5;
        }
    }

public:

    // iterator constructor [12]
    TernaryTreePrefixIterator( const TTree* aTTree ): fTTree(aTTree), fStack()
    {
        if (!(*fTTree).empty())
        {
            fStack.push(const_cast<TTreeNode>(fTTree));
        }
    }

    // iterator dereference [8]
    const T& operator*() const
    {
        return **fStack.top();
    }

    // prefix increment [12]
    Iterator& operator++()
    {
        TTreeNode lPopped = const_cast<TTreeNode>(fStack.top());
        fStack.pop();
        push_subtrees(lPopped);
        return *this;
    }

    // iterator equivalence [12]
    bool operator==( const Iterator& aOtherIter ) const
    {
        return fTTree == aOtherIter.fTTree && fStack.size() == aOtherIter.fStack.size();
    }

    // auxiliaries [4,10]
    Iterator begin() const
    {
        Iterator temp = *this;
        temp.fStack = TTreeStack();
        temp.fStack.push(const_cast<TTreeNode>(temp.fTTree));
        return temp;
    }
    Iterator end() const
    {
        Iterator temp = *this;
        temp.fStack = TTreeStack();
        return temp;
    }
};

```