```
1
2  // COS30008, Problem Set 4, Problem 3, 2022
3
4  #pragma once
5
6  #include "BinarySearchTree.h"
7
8  #include <stack>
9
10 template<typename T>
11 class BinarySearchTreeIterator
12 {
13 private:
14
15     using BSTree = BinarySearchTree<T>;
16     using BNode = BinaryTreeNode<T>;
17     using BTreeNode = BNode*;
18     using BTNStack = std::stack<BTreeNode>;
19
20     const BSTree& fBSTree;      // binary search tree
21     BTNStack fStack;            // DFS traversal stack
22
23     // perform a DFS traversal along the left side of the tree
24     void pushLeft(BTreeNode aNode)
25     {
26         while (!aNode->empty())
27         {
28             fStack.push(aNode);
29             aNode = aNode->left;
30         }
31     }
32
33 public:
34
35     using Iterator = BinarySearchTreeIterator<T>;
36
37     // constructor
38     BinarySearchTreeIterator( const BSTree& aBSTree ) :
39         fBSTree(aBSTree),
40         fStack(BTNStack())
41     {
42         pushLeft(fBSTree.fRoot);
43     }
44
45     // dereference operator
46     const T& operator*() const
47     {
48         return fStack.top()->key;
49     }
```

```
50
51      // prefix increment
52      Iterator& operator++()
53      {
54          BTreeNode lNode = fStack.top();
55          fStack.pop();
56          pushLeft(lNode->right);
57          return *this;
58      }
59
60      // postfix increment
61      Iterator operator++(int)
62      {
63          Iterator lTemp = *this;
64          ++(*this);
65          return lTemp;
66      }
67
68      // comparison operators
69      bool operator==(const Iterator& aOtherIter) const
70      {
71          return (&fBSTree == &aOtherIter.fBSTree)
72              && (fStack == aOtherIter.fStack);
73      }
74
75      bool operator!=(const Iterator& aOtherIter) const
76      {
77          return !(*this == aOtherIter);
78      }
79
80      // return an iterator with initialized stack
81      Iterator begin() const
82      {
83          return Iterator(fBSTree);
84      }
85
86      // return an end iterator with empty stack
87      Iterator end() const
88      {
89          Iterator lIter = *this;
90          lIter.fStack = BTNStack();
91          return lIter;
92      }
93  };
94
```