```
1
2  // COS30008, Final Exam, 2023
3
4  #pragma once
5
6  #include <memory>
7  #include <cassert>
8  #include <algorithm>
9
10 template<typename T>
11 class TernaryTree
12 {
13 public:
14
15     using Node = std::unique_ptr<TernaryTree>;
16
17 public:
18
19     TernaryTree(const T& aKey = T{}) noexcept : fKey(aKey)
20
21     {
22         for (size_t i = 0; i < 3; i++) {
23             fNodes[i] = nullptr;
24         }
25     }
26     TernaryTree(T&& aKey) noexcept : fKey(std::move(aKey))
27     {
28         for (size_t i = 0; i < 3; i++) {
29             fNodes[i] = nullptr;
30         }
31     }
32
33     template<typename... Args>
34     static Node makeNode(Args&&... args) {
35         return std::make_unique<TernaryTree>(std::forward<Args>(args)...);
36     }
37
38     const T& operator*() const noexcept {
39         return fKey;
40     }
41
42     TernaryTree& operator[](size_t aIndex) const {
43         assert(aIndex < 3);
44         return *fNodes[aIndex];
45     }
46
47     void add(size_t aIndex, Node& aNode) {
48         assert(aIndex < 3);
49         if (this) {
```

```
50              if (!fNodes[aIndex]) {
51                  fNodes[aIndex] = std::move(aNode);
52              }
53          }
54      }
55      Node remove(size_t aIndex) {
56          assert(aIndex < 3);
57          if (this) {
58              if (fNodes[aIndex]) {
59                  Node lRemoved = std::move(fNodes[aIndex]);
60                  return lRemoved;
61              }
62          }
63      }
64
65      bool leaf() const noexcept {
66          if (this) {
67              for (size_t i = 0; i < 3; i++) {
68                  if (fNodes[i] == nullptr)  {
69                      continue;
70                  }
71                  else {
72                      return false;
73                  }
74              }
75          }
76          return true;
77      }
78      size_t height() const noexcept {
79          if (leaf()) {
80              return 0;
81          }
82          else {
83              size_t lHeight = fNodes[0]->height();
84              size_t mHeight = fNodes[1]->height();
85              size_t rHeight = fNodes[2]->height();
86
87              if (lHeight >= mHeight && lHeight >= rHeight) return lHeight+1;
88              if (mHeight >= lHeight && mHeight >= rHeight) return mHeight+1;
89              if (rHeight >= mHeight && rHeight >= lHeight) return rHeight+1;
90          }
91      }
92
93  private:
94
95      T fKey;
96      Node fNodes[3];
97  };
98
```