```
1  // COS30008, Final Exam, 2024
2
3  // DynamicQueue.h
4  #pragma once
5
6  #include <optional>
7  #include <cassert>
8
9  #include <iostream>
10
11 template<typename T>
12 class DynamicQueue
13 {
14 private:
15     T* fElements;
16     size_t fFirstIndex;
17     size_t fLastIndex;
18     size_t fCurrentSize;
19
20     void resize(size_t aNewSize) {
21         T* lNewElements = new T[aNewSize];
22         size_t j = 0;
23         for (size_t i = fFirstIndex; i < fLastIndex; ++i, ++j) {
24             lNewElements[j] = std::move(fElements[i]);
25         }
26         delete[] fElements;
27         fElements = lNewElements;
28         fFirstIndex = 0;
29         fLastIndex = j;
30         fCurrentSize = aNewSize;
31     }
32
33     void ensure_capacity() {
34         if (fLastIndex >= fCurrentSize) {
35             resize(fCurrentSize * 2);
36         }
37     }
38
39     void adjust_capacity() {
40         if ((fLastIndex - fFirstIndex) <= fCurrentSize / 4 && fCurrentSize
           > 1) {
41             resize(fCurrentSize / 2);
42         }
43     }
44
45
46 public:
47     DynamicQueue() : fElements(new T[1]), fFirstIndex(0), fLastIndex(0),
       fCurrentSize(1) {}
```

```
48
49      ~DynamicQueue() {
50          delete[] fElements;
51      }
52
53      DynamicQueue(const DynamicQueue&) = delete;
54      DynamicQueue& operator=(const DynamicQueue&) = delete;
55
56      std::optional<T> top() const noexcept {
57          if (fFirstIndex == fLastIndex) {
58              return std::nullopt;
59          }
60          return fElements[fFirstIndex];
61      }
62
63      void enqueue(const T& aValue) {
64          ensure_capacity();
65          fElements[fLastIndex++] = aValue;
66      }
67
68      void dequeue() {
69          if (fFirstIndex < fLastIndex) {
70              ++fFirstIndex;
71              adjust_capacity();
72          }
73      }
74
75  };
```