```cpp
// COS30008, Final Exam, 2024

#pragma once

#include <optional>
#include <cassert>

#include <iostream>

template<typename T>
class DynamicQueue
{
private:
    T* fElements;
    size_t fFirstIndex;
    size_t fLastIndex;
    size_t fCurrentSize;

    void resize(size_t aNewSize) {
        T* lNewElements = new T[aNewSize];
        size_t j = 0;
        for (size_t i = fFirstIndex; i < fLastIndex; ++i, ++j) {
            lNewElements[j] = std::move(fElements[i]);
        }
        delete[] fElements;
        fElements = lNewElements;
        fFirstIndex = 0;
        fLastIndex = j;
        fCurrentSize = aNewSize;
    }

    void ensure_capacity() {
        if (fLastIndex >= fCurrentSize) {
            resize(fCurrentSize * 2);
        }
    }

    void adjust_capacity() {
        if ((fLastIndex - fFirstIndex) <= fCurrentSize / 4 &&
          fCurrentSize > 1) {
            resize(fCurrentSize / 2);
        }
    }


public:
    DynamicQueue() : fElements(new T[1]), fFirstIndex(0), fLastIndex(0),
      fCurrentSize(1) {}

    ~DynamicQueue() {
        delete[] fElements;
    }

    DynamicQueue(const DynamicQueue&) = delete;
    DynamicQueue& operator=(const DynamicQueue&) = delete;

    std::optional<T> top() const noexcept {
        if (fFirstIndex == fLastIndex) {
            return std::nullopt;
        }
        return fElements[fFirstIndex];
    }

    void enqueue(const T& aValue) {
        ensure_capacity();
        fElements[fLastIndex++] = aValue;
    }

    void dequeue() {
        if (fFirstIndex < fLastIndex) {
            ++fFirstIndex;
            adjust_capacity();
        }
    }

};
```