Swinburne University of Technology

Faculty of Science, Engineering and Technology

MIDTERM COVER SHEET

Your name:	Your student ID:		
Lecturer:	Dr. Markus Lumpe		
Due date:	April 26, 2024, 10:30		
Assignment number and title:	Midterm: Solution Design & Iterators		
Subject Title:	Data Structures and Patterns		
Subject Code:	COS30008		

Problem	Marks	Obtained
1	106	
2	194	
Total	300	

```
1 #include "KeyProvider.h"
2 #include <cctype>
3 #include <cassert>
   std::string KeyProvider::preprocessString(const std::string& aString)
     noexcept {
 6
       std::string result;
7
       for (char c : aString)
8
9
            if (isalpha(c))
10
            {
               result += toupper(c);
11
12
            }
       }
13
14
       return result;
15 }
16
17 KeyProvider::KeyProvider(const std::string& aKeyword, const std::string& →
      aSource) noexcept {
18
       std::string lKeyword = preprocessString(aKeyword);
       std::string lSource = preprocessString(aSource);
19
       for (size_t i = 0; i < lSource.length(); i++)</pre>
20
21
22
           fKeys += lKeyword[i % lKeyword.length()];
       }
23
24
       fIndex = 0;
25
       assert(fKeys.length() == lSource.length());
26 }
27
28 char KeyProvider::operator*() const noexcept {
29
       return fKeys[fIndex];
30 }
31
32 KeyProvider& KeyProvider::operator++() noexcept {
33
       fIndex++;
34
       return *this;
35 }
36
37 KeyProvider KeyProvider::operator++(int) noexcept {
       KeyProvider old = *this;
39
       ++(*this);
40
       return old;
41 }
42
43 bool KeyProvider::operator==(const KeyProvider& aOther) const noexcept {
44
       return fIndex == a0ther.fIndex && fKeys == a0ther.fKeys;
45 }
46
47 bool KeyProvider::operator!=(const KeyProvider& aOther) const noexcept {
       return !(*this == a0ther);
48
49 }
50
51 KeyProvider KeyProvider::begin() const noexcept {
```

```
D:\COS30008\Programs\midterm\KeyProvider.cpp
```

```
52
       KeyProvider temp = *this;
53
       temp.fIndex = 0;
54
       return temp;
55 }
56
57 KeyProvider KeyProvider::end() const noexcept {
       KeyProvider temp = *this;
       temp.fIndex = fKeys.size();
59
60
       return temp;
61 }
62
```

2

```
1 #include <cctype>
 2 #include "VigenereForwardIterator.h"
 3
 4 VigenereForwardIterator::VigenereForwardIterator(const std::string&
     aKeyword, const std::string& aSource, EVigenereMode aMode) noexcept :
 5
       fMode(aMode),
       fKeys(aKeyword, aSource),
 6
 7
       fSource(aSource),
 8
       fIndex(-1),
 9
       fCurrentChar('\0')
10 {
       initializeTable();
11
12 }
13
14 void VigenereForwardIterator::encodeCurrentChar() noexcept {
        char sourceChar = fSource[fIndex];
15
16
        if (std::isalpha(sourceChar))
17
            char keywordChar = std::toupper(*fKeys);
18
19
            size_t row = keywordChar - 'A';
            size_t col = std::toupper(sourceChar) - 'A';
20
            char encodedChar = fMappingTable[row][col];
21
            if (std::islower(sourceChar)) {
22
                fCurrentChar = std::tolower(encodedChar);
23
            }
24
            else {
25
                fCurrentChar = encodedChar;
26
27
28
            fKeys++;
29
       }
       else {
30
31
            fCurrentChar = sourceChar;
32
33 }
34
   void VigenereForwardIterator::decodeCurrentChar() noexcept {
35
        char sourceChar = fSource[fIndex];
36
37
        if (std::isalpha(sourceChar))
38
39
            char keywordChar = std::toupper(*fKeys);
40
            size_t row = keywordChar - 'A';
            for (size_t i = 0; i < CHARACTERS; ++i) {</pre>
41
42
                if (fMappingTable[row][i] == std::toupper(sourceChar)) {
43
                    char decodedChar = 'A' + i;
                    if (std::islower(sourceChar)) {
44
45
                        fCurrentChar = std::tolower(decodedChar);
                    }
46
47
                    else {
48
                        fCurrentChar = decodedChar;
                    ş
49
                    break;
50
51
                }
            }
52
```

```
D:\COS30008\Programs\midterm\VigenereForwardIterator.cpp
```

```
2
```

```
53
            fKeys++;
54
        }
55
        else {
            fCurrentChar = sourceChar;
56
 57
58 }
59
60
61 char VigenereForwardIterator::operator*() const noexcept {
        return fCurrentChar;
62
63 }
64
65
    VigenereForwardIterator& VigenereForwardIterator::operator++() noexcept →
66
        fIndex++;
        if (fIndex < fSource.size())</pre>
67
68
69
             if (fMode == EVigenereMode::Encode) {
70
                 encodeCurrentChar();
            }
71
            else {
72
73
                 decodeCurrentChar();
74
            }
75
        }
76
        return *this;
77 }
78
79 VigenereForwardIterator VigenereForwardIterator::operator++(int)
      noexcept {
80
        VigenereForwardIterator old = *this;
        ++(*this);
81
82
        return old;
83 }
84
    bool VigenereForwardIterator::operator==(const VigenereForwardIterator& →
       aOther) const noexcept {
        return fIndex == a0ther.fIndex && fSource == a0ther.fSource;
86
87 }
88
89 bool VigenereForwardIterator::operator!=(const VigenereForwardIterator& >
       aOther) const noexcept {
        return !(*this == a0ther);
90
91 }
92
    VigenereForwardIterator VigenereForwardIterator::begin() const noexcept >
93
        VigenereForwardIterator result = *this;
94
95
        if (result.fIndex < result.fSource.size())</pre>
96
            if (result.fMode == EVigenereMode::Encode) {
97
                 result.encodeCurrentChar();
98
99
            }
            else {
100
```

```
D:\COS30008\Programs\midterm\VigenereForwardIterator.cpp
```

```
101
                result.decodeCurrentChar();
102
            }
103
        }
104
        return result;
105 }
106
107 VigenereForwardIterator VigenereForwardIterator::end() const noexcept {
108
        VigenereForwardIterator result = *this;
109
        result.fIndex = fSource.size();
110
        return result;
111 }
112
```