

**Swinburne University of Technology***Faculty of Science, Engineering and Technology***MIDTERM COVER SHEET**

---

**Subject Code:**

COS30008

**Subject Title:**

Data Structures and Patterns

**Assignment number and title:**

Midterm: Solution Design &amp; Iterators

**Due date:**

April 26, 2024, 10:30

**Lecturer:**Dr. Markus Lumpe

---

**Your name:** \_\_\_\_\_ **Your student ID:** \_\_\_\_\_

---

Marker's comments:

Problem	Marks	Obtained
1	106	
2	194	
Total	300	

---

```
1 // Cos3008-Mid
2 //Created By NUR E SIAM
3
4
5 #include <stdio.h>
6 #include "KeyProvider.h"
7 #include <cctype>
8 #include <cassert>
9
10 KeyProvider::KeyProvider(const std::string& aKeyword, const std::string& aSource) noexcept
11 {
12     fIndex = 0;
13     std::string Ckeyword = preprocessString(aKeyword);
14     std::string Csource = preprocessString(aSource);
15
16     // goes through the length of preprocessed aSource, and for each letter, applies a preprocessed aKeyword letter
17     for (size_t i = 0; i < Csource.length(); i++)
18     {
19         fKeys += Ckeyword[i % Ckeyword.length()];
20     }
21
22     assert(fKeys.length() == Csource.length());
23 }
24
25 std::string KeyProvider::preprocessString(const std::string& aString) noexcept
26 {
27     // For each loop that goes through each letter of aString
28     // Checks that it complies to isalpha()
29     // make it uppercase with toupper()
30     // return string
31
32     std::string pString;
33     for (char c : aString)
34     {
35         if (isalpha(c))
36         {
37             pString += toupper(c);
38         }
39     }
40     return pString;
41 }
42
43 char KeyProvider::operator*() const noexcept
44 {
45     return fKeys[fIndex];
46 }
```

```
47
48 KeyProvider& KeyProvider::operator++() noexcept
49 {
50     ++fIndex;
51     return *this;
52 }
53
54 KeyProvider KeyProvider::operator++(int) noexcept
55 {
56     // Return's a copy before advancing
57     KeyProvider temp = *this;
58     ++(*this);
59     return temp;
60 }
61
62 bool KeyProvider::operator==(const KeyProvider& aOther) const noexcept
63 {
64     return fIndex == aOther.fIndex && fKeys == aOther.fKeys;
65 }
66
67 bool KeyProvider::operator!=(const KeyProvider& aOther) const noexcept
68 {
69     return !(*this == aOther);
70 }
71
72 KeyProvider KeyProvider::begin() const noexcept
73 {
74     KeyProvider temp(*this); // Creates a copy
75     temp.fIndex = 0;
76     return temp; // Return's the old copy
77 }
78
79 KeyProvider KeyProvider::end() const noexcept
80 {
81     KeyProvider temp(*this);
82     temp.fIndex = fKeys.length();
83     return temp;
84 }
85
```

```
1 // Cos3008-Mid
2 //Created By NUR E SIAM
3
4
5 #include <cctype>
6 #include "VigenereForwardIterator.h"
7
8 VigenereForwardIterator::VigenereForwardIterator(const std::string&      ↵
9     aKeyword, const std::string& aSource, EVigenereMode aMode) noexcept :    ↵
10    fMode(aMode),
11    fKeys(aKeyword, aSource),
12    fSource(aSource),
13    fIndex(-1),
14    fCurrentChar('\0')
15 {
16     initializeTable();
17 }
18
19 char VigenereForwardIterator::operator*() const noexcept
20 {
21     return fCurrentChar;
22 }
23
24 VigenereForwardIterator& VigenereForwardIterator::operator++() noexcept
25 {
26     ++fIndex;
27     if (fIndex >= fSource.length())
28     {
29         return *this;
30     }
31     if (std::isalpha(fSource[fIndex]))
32     {
33         if (fMode == EVigenereMode::Encode)
34             encodeCurrentChar();
35         else
36             decodeCurrentChar();
37         ++fKeys;
38     }
39     else
40     {
41         fCurrentChar = fSource[fIndex]; // Non-alphabetic characters      ↵
42         remain unchanged
43     }
44     return *this;
45 }
46
```

```
47 VigenereForwardIterator VigenereForwardIterator::operator++(int) noexcept
48 {
49     VigenereForwardIterator temp = *this;
50     ++(*this);
51     return temp;
52 }
53
54 bool VigenereForwardIterator::operator==(const VigenereForwardIterator& aOther) const noexcept
55 {
56     return (fIndex == aOther.fIndex);
57 }
58
59 bool VigenereForwardIterator::operator!=(const VigenereForwardIterator& aOther) const noexcept
60 {
61     return !(*this == aOther);
62 }
63
64 VigenereForwardIterator VigenereForwardIterator::begin() const noexcept
65 {
66     return *this;
67 }
68
69 VigenereForwardIterator VigenereForwardIterator::end() const noexcept
70 {
71     VigenereForwardIterator res = *this;
72     res.fIndex = fSource.length();
73     return res;
74 }
75
76 void VigenereForwardIterator::encodeCurrentChar() noexcept
77 {
78     char sourceCharacter = fSource[fIndex];
79     char keyCharacter = std::toupper(*fKeys);
80
81     if (std::isalpha(sourceCharacter))
82     {
83         char encodedCharacter = fMappingTable[std::toupper(keyCharacter) - 'A'][std::toupper(sourceCharacter) - 'A'];
84         if (std::islower(sourceCharacter))
85         {
86             encodedCharacter = std::tolower(encodedCharacter);
87         }
88         fCurrentChar = encodedCharacter;
89     }
90     else
91     {
92         fCurrentChar = sourceCharacter;
```

```
93     }
94 }
95
96 void VigenereForwardIterator::decodeCurrentChar() noexcept
97 {
98     char sourceCharacter = fSource[fIndex];
99     char keyCharacter = std::toupper(*fKeys);
100
101    if (std::isalpha(sourceCharacter))
102    {
103        char decodedCharacter = 'A';
104        if (std::isupper(sourceCharacter))
105        {
106            for (int i = 0; i < CHARACTERS; ++i)
107            {
108                if (fMappingTable[keyCharacter - 'A'][i] == std::toupper
109                    (sourceCharacter))
110                {
111                    decodedCharacter = 'A' + i;
112                    break;
113                }
114            }
115        else if (std::islower(sourceCharacter))
116        {
117            for (int i = 0; i < CHARACTERS; ++i)
118            {
119                if (fMappingTable[keyCharacter - 'A'][i] == std::toupper
120                    (sourceCharacter))
121                {
122                    decodedCharacter = 'a' + i;
123                    break;
124                }
125            }
126        fCurrentChar = decodedCharacter;
127    }
128 else
129 {
130    fCurrentChar = sourceCharacter;
131 }
132 }
```