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, Design Pattern, and Iterators
Due date: April 27, 2022, 23:59
Lecturer: Dr. Markus Lumpe

Your name: Trac Duc Anh Luong **Your student ID:** 103488117

Check Tutorial	Mon 10:30	Mon 14:30	Tues 08:30	Tues 10:30	Tues 12:30	Tues 14:30	Tues 16:30	Wed 08:30	Wed 10:30	Wed 12:30	Wed 14:30
				X							

Marker's comments:

Problem	Marks	Obtained
1	68	
2	120	
3	56	
4	70	
Total	314	

```cpp
#include "KeyProvider.h"

KeyProvider::KeyProvider(const std::string& aKeyword): fKeyword(new char
  [aKeyword.length()]), fSize(aKeyword.length()), fIndex(0)
{
    initialize(aKeyword);
}

KeyProvider::~KeyProvider()
{
    delete[] fKeyword;
}

void KeyProvider::initialize(const std::string& aKeyword)
{
    delete[] fKeyword;
    fSize = aKeyword.length();
    fKeyword = new char[fSize];
    for (size_t i = 0; i < fSize; i++)
    {
        fKeyword[i] = static_cast<char>(toupper(aKeyword[i]));
    }
    fIndex = 0;
}

char KeyProvider::operator*() const
{
    return fKeyword[fIndex];
}

KeyProvider& KeyProvider::operator<<(char aKeyCharacter)
{
    fKeyword[fIndex] = static_cast<char>(toupper(aKeyCharacter));
    if (++fIndex >= fSize)
    {
        fIndex = 0;
    }
    return *this;
}
```

```cpp
#include "Vigenere.h"

void Vigenere::initializeTable()
{
    for (char row = 0; row < CHARACTERS; row++)
    {
        char lChar = 'B' + row;
        for (char column = 0; column < CHARACTERS; column++)
        {
            if (lChar > 'Z')
                lChar = 'A';
            fMappingTable[row][column] = lChar++;
        }
    }
}

Vigenere::Vigenere(const std::string& aKeyword): fKeyword(aKeyword), fKeywordProvider(KeyProvider(aKeyword))
{
    initializeTable();
}

std::string Vigenere::getCurrentKeyword()
{
    std::string current_keyword;

    for (size_t i = 0; i < fKeyword.length(); i++)
    {
        current_keyword += *fKeywordProvider;
        fKeywordProvider << *fKeywordProvider;
    }
    return current_keyword;
}

void Vigenere::reset()
{
    fKeywordProvider.initialize(fKeyword);
}

char Vigenere::encode(char aCharacter)
{
    if (isalpha(aCharacter))
    {
        bool isLower = std::islower(aCharacter);
        char encoded = fMappingTable[*fKeywordProvider - 'A'][std::toupper(aCharacter) - 'A'];

        fKeywordProvider << aCharacter;
        if (isLower)
```

```
48          {
49              return static_cast<char>(std::tolower(encoded));
50          }
51          return encoded;
52      }
53      return aCharacter;
54  }
55
56  char Vigenere::decode(char aCharacter)
57  {
58      if (isalpha((aCharacter)))
59      {
60          bool isLower = std::islower(aCharacter);
61          char encoded = static_cast<char>(toupper(aCharacter));
62          char decoded = 0;
63
64          for (char column = 0; column < CHARACTERS; column++)
65          {
66              if (fMappingTable[*fKeywordProvider - 'A'][column] == encoded)
67              {
68                  decoded = static_cast<char>(column + 'A');
69                  break;
70              }
71          }
72
73          fKeywordProvider << decoded;
74          if (isLower)
75          {
76              return static_cast<char>(std::tolower(decoded));
77          }
78          return decoded;
79      }
80      return aCharacter;
81  }
```

```cpp
#include "iVigenereStream.h"

iVigenereStream::iVigenereStream(Cipher aCipher, const std::string& aKeyword, const char* aFileName): fIStream(std::ifstream()), fCipherProvider(Vigenere(aKeyword)), fCipher(std::move(aCipher))
{
    if (aFileName != nullptr)
    {
        open(aFileName);
    }
}

iVigenereStream::~iVigenereStream()
{
    close();
}

void iVigenereStream::open(const char* aFileName)
{
    fIStream.open(aFileName, std::ios::binary);
}

void iVigenereStream::close()
{
    fIStream.close();
}

void iVigenereStream::reset()
{
    fCipherProvider.reset();
    seekstart();
}

bool iVigenereStream::good() const
{
    return fIStream.good();
}

bool iVigenereStream::is_open() const
{
    return fIStream.is_open();
}

bool iVigenereStream::eof() const
{
    return fIStream.eof();
}

iVigenereStream& iVigenereStream::operator>>(char& aCharacter)
```

```
48  {
49      aCharacter = fCipher(fCipherProvider, static_cast<char>(fIStream.get
          ()));
50      return *this;
51  }
```

```cpp
#include "VigenereForwardIterator.h"

VigenereForwardIterator::VigenereForwardIterator(iVigenereStream&
  aIStream): fIStream(aIStream), fCurrentChar(0), fEOF(aIStream.eof())
{
    if (!fEOF)
    {
        fIStream >> fCurrentChar;
    }
}

char VigenereForwardIterator::operator*() const
{
    return fCurrentChar;
}

VigenereForwardIterator& VigenereForwardIterator::operator++()
{
    fIStream >> fCurrentChar;
    fEOF = fIStream.eof();
    return *this;
}

VigenereForwardIterator VigenereForwardIterator::operator++(int)
{
    VigenereForwardIterator temp = *this;
    ++(*this);
    return temp;
}

bool VigenereForwardIterator::operator==(const VigenereForwardIterator&
  aOther) const
{
    return (&fIStream == &aOther.fIStream) && (fEOF == aOther.fEOF);
}

bool VigenereForwardIterator::operator!=(const VigenereForwardIterator&
  aOther) const
{
    return !(*this == aOther);
}

VigenereForwardIterator VigenereForwardIterator::begin() const
{
    VigenereForwardIterator lResult = *this;
    lResult.fIStream.reset();
    lResult.fEOF = lResult.fIStream.eof();
    if (!lResult.fEOF)
    {
```

```
47          lResult.fIStream >> lResult.fCurrentChar;
48      }
49      return lResult;
50  }
51
52  VigenereForwardIterator VigenereForwardIterator::end() const
53  {
54      VigenereForwardIterator lResult = *this;
55      lResult.fEOF = true;
56      return lResult;
57  }
```