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: Ta Quang Tung **Your student ID:** 104222196

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

Marker's comments:

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

KeyProvider.cpp

```cpp
#include "KeyProvider.h"
#include <cstring>

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

    for (size_t i = 0; i < fSize; i++) {
        fKeyword[i] = toupper(aKeyword[i]);
    }
}

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

void KeyProvider::initialize(const std::string& aKeyword) {
    // If the length of the new keyword is the same as the old one, there is no need to
delete the char array.
    if (fSize != aKeyword.length()) {
        delete[] fKeyword;
        fSize = aKeyword.length();
        fKeyword = new char[fSize];
    }

    fIndex = 0;

    for (size_t i = 0; i < fSize; i++) {
        fKeyword[i] = toupper(aKeyword[i]);
    }
}

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

KeyProvider& KeyProvider::operator<<(char aKeyCharacter) {
    fKeyword[fIndex] = toupper(aKeyCharacter);

    fIndex++;
    if (fIndex >= fSize) {
        fIndex = 0;
    }

    return *this;
}
```

Vigenere.cpp

```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 keyword = "";

    size_t length = fKeyword.length();

    for (size_t i = 0; i < length; i++) {
        char c = *fKeywordProvider;
        keyword += c;
        fKeywordProvider << c;
    }

    return keyword;
}

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

char Vigenere::encode(char aCharacter) {
    bool isLower = islower(aCharacter);

    char currentKeyChar = *fKeywordProvider;
    char charToEncode = toupper(aCharacter);

    char encoded;

    if (isalpha(currentKeyChar) && isalpha(charToEncode)) {
        encoded = fMappingTable[currentKeyChar - 'A'][charToEncode - 'A'];
        fKeywordProvider << charToEncode;
    } else {
        encoded = charToEncode;
    }

    if (isLower) {
        return tolower(encoded);
    } else {
        return encoded;
    }
}

char Vigenere::decode(char aCharacter) {
    bool isLower = islower(aCharacter);

    char currentKeyChar = *fKeywordProvider;
    char charToDecode = toupper(aCharacter);

    char decoded = charToDecode;

    if (isalpha(currentKeyChar) && isalpha(charToDecode)) {
        size_t row = currentKeyChar - 'A';

        for (size_t col = 0; col < CHARACTERS; col++) {
            if (charToDecode == fMappingTable[row][col]) {
                decoded = col + 'A';
                break;
            }
        }

        fKeywordProvider << decoded;
    }

    if (isLower) {
        return tolower(decoded);
    } else {
        return decoded;
    }
}
```

iVigenereStream.cpp

```cpp
#include "iVigenereStream.h"

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

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

void iVigenereStream::open(const char* aFileName) {
    fIStream.open(aFileName, std::ifstream::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) {
    char c = fIStream.get();
    aCharacter = fCipher(fCipherProvider, c);
    return *this;
}
```

VigenereForwardIterator.cpp

```cpp
#include "VigenereForwardIterator.h"

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

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

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

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

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 begin = *this;
    begin.fIStream.reset();
    begin.fEOF = begin.fIStream.eof();
    begin.fIStream >> begin.fCurrentChar;

    return begin;
}

VigenereForwardIterator VigenereForwardIterator::end() const {
    VigenereForwardIterator end = *this;
    end.fEOF = true;
    return end;
}
```