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: Jamie Kozminska **Your student ID:** 101114436

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"
using namespace std;

KeyProvider::KeyProvider(const std::string& aKeyword) :
    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++)
    {
        this->fKeyword[i] = toupper(aKeyword.at(i));
    }

    fIndex = 0;
}

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

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


```

```cpp

// COS30008, Midterm, Problem 2, 2022

#include "Vigenere.h"

using namespace std;

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(aKeyword)
{
    fMappingTable[CHARACTERS][CHARACTERS];
    initializeTable();
}

std::string Vigenere::getCurrentKeyword()
{
    string result;
    for (size_t i = 0; i < fKeyword.length(); i++)
    {
        char temp = *fKeywordProvider;
        fKeywordProvider << temp;
        result += fKeyword[i];
    }
    return result;
}

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

char Vigenere::encode(char aCharacter)
```

```
50  {
51
52      if (!isalpha(aCharacter)) {
53          return aCharacter;
54      }
55
56      char rawKey = *fKeywordProvider;
57      char key = toupper(rawKey) - 'A' + 1;
58      char lCharacter = toupper(aCharacter) - 'A' - 1;
59      char result = fMappingTable[key][lCharacter];
60      fKeywordProvider << aCharacter;
61
62      if (isupper(aCharacter)) {
63          return toupper(result);
64      }
65      return tolower(result);
66  }
67
68  char Vigenere::decode(char aCharacter)
69  {
70      if (!isalpha(aCharacter)) {
71          return aCharacter;
72      }
73
74      char rawKey = *fKeywordProvider;
75      char key = toupper(rawKey) - 'A';
76      char encodedChar = toupper(aCharacter);
77
78      char result;
79      for (size_t i = 0; i < CHARACTERS; i++) {
80
81          char columnChar = fMappingTable[key][i];
82
83          if (encodedChar == columnChar) {
84              result = i + 'A';
85              break;
86          }
87      }
88      fKeywordProvider << result;
89
90      if (isupper(aCharacter)) {
91          return toupper(result);
92      }
93      return tolower(result);
94  }
95
```

```cpp
#include "Vigenere.h"
#include "iVigenereStream.h"
#include <fstream>
using namespace std;


iVigenereStream::iVigenereStream(Cipher aCipher, const std::string& aKeyword,
  const char* aFileName) :
    fCipherProvider(aKeyword)
{
    fCipher = aCipher;

    if (!nullptr)
    {
        fIStream.open(aFileName);
    }
}

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

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

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

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

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

bool iVigenereStream::is_open() const
{
```

```
49      return fIStream.is_open();
50  }
51
52  bool iVigenereStream::eof() const
53  {
54      return fIStream.eof();
55  }
56
57  iVigenereStream& iVigenereStream::operator>>(char& aCharacter)
58  {
59      char temp;
60      if (fCipher.operator bool() && fIStream.get(temp))
61      {
62          aCharacter = fCipher(fCipherProvider, temp);
63      }
64      return *this;
65  }
66
```

```cpp
#include "VigenereForwardIterator.h"

VigenereForwardIterator::VigenereForwardIterator(iVigenereStream& aIStream) :
    fIStream(aIStream),
    fCurrentChar(0),
    fEOF(false)
{
    fIStream.seekstart();

    aIStream >> fCurrentChar;
}

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

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

VigenereForwardIterator VigenereForwardIterator::operator++(int)
{
    VigenereForwardIterator result = *this;
    if (fEOF == false)
    {
        ++(*this);
    }
    return result;
}

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

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

```

```
48 VigenereForwardIterator VigenereForwardIterator::begin() const
49 {
50     VigenereForwardIterator result = *this;
51     result.fIStream.reset();
52     return result;
53 }
54
55 VigenereForwardIterator VigenereForwardIterator::end() const
56 {
57     VigenereForwardIterator Result = *this;
58     Result.fEOF = true;
59     return Result;
60 }
61
```