**Objective:**

* To understand the concept of polymorphism in C++.

**Challenge - 1:** Encryptor App  **(**15**)**

Imagine you are working as a cybersecurity engineer at a leading technology company, tasked with developing a secure messaging feature for a new communication platform. The platform aims to provide end-to-end encryption for messages exchanged between users to ensure confidentiality and privacy.

Your task is to design and implement a program that can encrypt and decrypt messages using various encryption algorithms. The program should include the following classes:

**Encryptor:** This is an abstract base class representing a generic encryption algorithm.

* virtual String encrypt(const String& message) const = 0;
* virtual String decrypt(const String& cipher) const = 0;

We are going to Discuss/Implement two Different cipher techniques named Cease Cipher and Vignere Cipher and going to develop a class for each of them.

| **CaesarCipher** | This class inherits from the Encryptor class. The Caesar cipher shifts each letter in the message by a fixed number of positions in the alphabet. |
| --- | --- |
| **Encryption Process:** | Each letter in the message will be replaced by the letter located a fixed number of positions forward in the alphabet. |
|  | If the shift is 3, 'A' will be replaced by 'D', 'B' will be replaced by 'E', if ‘Z’ it will round off to ‘C’, and so on. |
| **Example:** | "HELLO" encrypted with a Caesar Cipher with a shift of 3 becomes "KHOOR".  H Shifts to K → H,I,J then K. H after shift of 3 becomes K. and so on. |
| **Decryption Process:** | To decrypt the message, we'll reverse the process by replacing each letter in the ciphertext with the letter located a fixed number of positions backward in the alphabet. |
| **Example:** | "KHOOR" decrypted with a Caesar Cipher with a shift of 3 becomes "HELLO".  K Shifts back to H → K,J,I then H. K after shift of 3 becomes H. and so on. |

| **VigenereCipher** | This class inherits from the Encryptor class. |
| --- | --- |
| **Encryption Process:** | Vigenere Cipher is a method of encrypting alphabetic text by using a keyword and a repeating sequence of different Caesar ciphers. |
| **Example:** | Let's say we have the keyword "KEY" and the plaintext "HELLO". We align the keyword with the plaintext: |
| | **Plaintext:** | H | E | L | L | O | | --- | --- | --- | --- | --- | --- |  | **Keyword:** | K | E | Y | K | E | | --- | --- | --- | --- | --- | --- | | **Shift Value:** | 10 | 4 | 24 | 10 | 4 |   e.g. A has a shift value of 0. B has a shift value of 1 and so on.  If a keyword is shorter than the plain text it will repeat itself.   | H | is shifted by 10 positions (K) | R | | --- | --- | --- | | E | is shifted by 4 positions (E) | I | | L | is shifted by 24 positions (Y) | J | | L | is shifted by 10 positions (K) | V | | O | is shifted by 4 positions (E) | S |   So, the resulting ciphertext is "RIJVS". | |
| **Decryption Process:** | Decryption in Vigenere Cipher involves using the same keyword to reverse the encryption process. Each letter in the ciphertext is shifted back according to the corresponding letter in the keyword to reveal the original plaintext. |
| **Example:** | Now we have the keyword "KEY" we’ll decrypt the word “RIJVS” |
| | **Encrypted Text:** | R | I | J | V | S | | --- | --- | --- | --- | --- | --- |  | **Keyword:** | K | E | Y | K | E | | --- | --- | --- | --- | --- | --- | | **Shift Value:** | 10 | 4 | 24 | 10 | 4 |   If a keyword is shorter than the plain text it will repeat itself.   | R | is shifted back by 10 positions (K) | H | | --- | --- | --- | | I | is shifted back by 4 positions (E) | E | | J | is shifted back by 24 positions (Y) | L | | V | is shifted back by 10 positions (K) | L | | S | is shifted back by 4 positions (E) | O |   So, the resulting decrypted text is "HELLO". | |

| **Encryptor** |
| --- |
| class Encryptor  {  public:  virtual String encrypt(const String& message) const = 0;  virtual String decrypt(const String& cipher) const = 0;  }; |

| **CaesarCipher** | **VigenereCipher** |
| --- | --- |
| class CaesarCipher : public Encryptor  {  private:  int shift;  public:  CaesarCipher(int shift);  void setShift(int newShift);  String encrypt(const String& message) const override;  String decrypt(const String& message) const override;  }; | class VigenereCipher : public Encryptor  {  private:  String keyword;  public:  VigenereCipher(const String& keyword);  void setKeyword(const String&newKeyword);  String encrypt(const String& message) const override;  String decrypt(const String& message) const override;  }; |

**Solution:**

**Encryptor.h**

#ifndef ENCRYPTOR\_H

#define ENCRYPTOR\_H

#include "String.h"

class Encryptor

{

public:

virtual String encrypt(const String& message) const = 0;

virtual String decrypt(const String& cipher) const = 0;

};

#endif // ENCRYPTOR\_H

**CaesarCipher.h**

#ifndef CAESARCIPHER\_H

#define CAESARCIPHER\_H

#include"Encryptor.h"

class CaesarCipher : public Encryptor

{

private:

int shift;

bool isAlphaChar(char ch) const;

char shiftChar(char ch, int shift) const;

public:

CaesarCipher(int shift);

void setShift(int newShift);

String encrypt(const String& message) const override;

String decrypt(const String& message) const override;

};

#endif // CAESARCIPHER\_H

**CaesarCipher.cpp ----- (5.5 Marks)**

#include "CaesarCipher.h"

**//Private Functions:**

bool CaesarCipher::isAlphaChar(char ch) const

{

return (ch>= 'A' &&ch<= 'Z') || (ch>= 'a' &&ch<= 'z');

}

char CaesarCipher::shiftChar(char ch, int shift) const

{

char base = (ch>= 'A' &&ch<= 'Z') ? 'A' : 'a';

return static\_cast<char>((ch - base + shift) % 26 + base);

}

**//Public Functions:**

CaesarCipher::CaesarCipher(int shift) : shift(shift) {}

void CaesarCipher::setShift(int newShift) —------------- **0.5**

{

shift = newShift;

}

String CaesarCipher::encrypt(const String& message) const —------------- **2.5**

{

String encryptedMessage = message;

for (int i = 0; i<encryptedMessage.getLength(); i++)

{

char&ch = encryptedMessage[i];

if (isAlphaChar(ch))

{

ch = shiftChar(ch, shift);

}

}

return encryptedMessage;

}

**Sample Runs:**

**Shift:3**

Original message: Good Morning

Encrypted message: Jrrg Pruqlqj —------------- **1**

**Shift:4**

Original message: Hurray! It's Monday!

Encrypted message: Lyvvec! Mx'w Qsrhec! —------------- **1**

**Shift:7**

Original message: How are you

Encrypted message: Ovd hyl fvb —------------- **0.5**

**-1 For automicity.**

**If the case is not maintained -1.**

String CaesarCipher::decrypt(const String& cipher) const —------------- **2.5**

{

String decryptedMessage = cipher;

for (int i = 0; i<decryptedMessage.getLength(); i++)

{

char&ch = decryptedMessage[i];

if (isAlphaChar(ch))

{

ch = shiftChar(ch, -shift);

}

}

return decryptedMessage;

}

**Sample Runs:**

**Shift:3**

Encrypted message: Jrrg Pruqlqj

Decrypted message: Good Morning —------------- **1**

**Shift:4**

Encrypted message: Lyvvec! Mx'w Qsrhec!

Decrypted message: Hurray! It's Monday! —------------- **1**

**Shift:7**

Encrypted message: Ovd hyl fvb

Decrypted message: How are you —------------- **0.5**

**-1 For automicity.**

**If the case is not maintained -1.**

**VigenereCipher.h**

#ifndef VIGENERECIPHER\_H

#define VIGENERECIPHER\_H

#include"Encryptor.h"

class VigenereCipher : public Encryptor

{

private:

String keyword;

bool isAlphaChar(char ch) const;

char shiftChar(char ch, int shift) const;

public:

VigenereCipher(const String& keyword);

void setKeyword(const String&newKeyword);

String encrypt(const String& message) const override;

String decrypt(const String& message) const override;

};

#endif // VIGENERECIPHER\_H

**VigenereCipher.cpp ----- (9.5 Marks)**

#include "VigenereCipher.h"

**//Private Functions:**

bool VigenereCipher::isAlphaChar(char ch) const

{

return (ch>= 'A' &&ch<= 'Z') || (ch>= 'a' &&ch<= 'z');

}

char VigenereCipher::shiftChar(char ch, int shift) const

{

char base = (ch>= 'A' &&ch<= 'Z') ? 'A' : 'a';

return static\_cast<char>((ch - base + shift) % 26 + base);

}

**//Public Functions:**

VigenereCipher::VigenereCipher(const String& keyword) : keyword(keyword) {}

void VigenereCipher::setKeyword(const String&newKeyword) —------------- **0.5**

{

keyword = newKeyword;

}

String VigenereCipher::encrypt(const String& message) const —------------- **4.5**

{

String encryptedMessage = message;

int keywordLength = keyword.getLength();

int keywordIndex = 0;

for (int i = 0; i<encryptedMessage.getLength(); i++)

{

char&ch = encryptedMessage[i];

if (isAlphaChar(ch))

{

char k = keyword[keywordIndex % keywordLength];

char base = (k>= 'A' &&k<= 'Z') ? 'A' : 'a;

ch = shiftChar(ch, k - base);

keywordIndex++;

}

}

return encryptedMessage;

}

**Sample Runs:**

**keyword: “KEY”**

Original message: Good Afternoon!

Encrypted message: Qsmn Eddipxsmx! —------------- **1.5**

**keyword: “OOP”**

Original message: Polypmorphism Lab!

Encrypted message: Dcamdbcfevwha Zpp! —------------- **1.5**

**keyword: “HELP”**

Original message: Wow

Encrypted message: Dsh —------------- **1.5**

**-1 For automicity.**

**If the case is not maintained -1. ie. Wow is encrypted to dsh as Wow should be encrypted to Dsh.**

**String VigenereCipher::decrypt(const String& message) const —------------- 4.5**

{

String decryptedMessage = message;

int keywordLength = keyword.getLength();

int keywordIndex = 0;

for (int i = 0; i<decryptedMessage.getLength(); i++)

{

char&ch = decryptedMessage[i];

if (isAlphaChar(ch))

{

char k = keyword[keywordIndex % keywordLength];

char base = (k>= 'A' &&k<= 'Z') ? 'A' : 'a;

ch = shiftChar(ch, -(k – base));

keywordIndex++;

}

}

return decryptedMessage;

}

**Sample Runs:**

**keyword: “KEY”**

Encrypted message: Qsmn Eddipxsmx!

Decrypted message: Good Afternoon! —------------- **1.5**

**keyword: “OOP”**

Encrypted message: Dcamdbcfevwha Zpp!

Decrypted message: Polypmorphism Lab! —------------- **1.5**

**keyword: “HELP”**

Encrypted message: Dsh

Decrypted message: Wow —------------- **1.5**

**-1 For automicity.**

**If the case is not maintained -1. ie. Dsh is decrypted to wow as Dsh should be decrypted to Wow.**

**Quick Revision:**

**CaesarCipher.cpp**

**void CaesarCipher::setShift(int newShift) —------------- 0.5**

**String CaesarCipher::encrypt(const String& message) const —----2.5**

**Sample Runs:**

**Shift:3**

Original message: Good Morning

Encrypted message: Jrrg Pruqlqj —------------- **1**

**Shift:4**

Original message: Hurray! It's Monday!

Encrypted message: Lyvvec! Mx'w Qsrhec! —------------- **1**

**Shift:7**

Original message: How are you

Encrypted message: Ovd hyl fvb —------------- **0.5**

**-1 For automicity.**

**If the case is not maintained -1.**

**String CaesarCipher::decrypt(const String& cipher) const ------- 2.5**

**Sample Runs:**

**Shift:3**

Encrypted message: Jrrg Pruqlqj

Decrypted message: Good Morning —------------- **1**

**Shift:4**

Encrypted message: Lyvvec! Mx'w Qsrhec!

Decrypted message: Hurray! It's Monday! —------------- **1**

**Shift:7**

Encrypted message: Ovd hyl fvb

Decrypted message: How are you —------------- **0.5**

**-1 For automicity.**

**If the case is not maintained -1.**

**VigenereCipher.cpp**

**void VigenereCipher::setKeyword(const String&newKeyword)—------- 0.5**

**String encrypt(const String& message) const override; —------------- 4.5**

**Sample Runs:**

**keyword: “KEY”**

Original message: Good Afternoon!

Encrypted message: Qsmn Eddipxsmx! —------------- **1.5**

**keyword: “OOP”**

Original message: Polypmorphism Lab!

Encrypted message: Dcamdbcfevwha Zpp! —------------- **1.5**

**keyword: “HELP”**

Original message: Wow

Encrypted message: Dsh —------------- **1.5**

**-1 For automicity.**

**If the case is not maintained -1. ie. Wow is encrypted to dsh as Wow should be encrypted to Dsh.**

**String decrypt(const String& message) const override; —------------- 4.5**

**Sample Runs:**

**keyword: “KEY”**

Encrypted message: Qsmn Eddipxsmx!

Decrypted message: Good Afternoon! —------------- **1.5**

**keyword: “OOP”**

Encrypted message: Dcamdbcfevwha Zpp!

Decrypted message: Polypmorphism Lab! —------------- **1.5**

**keyword: “HELP”**

Encrypted message: Dsh

Decrypted message: Wow —------------- **1.5**

**-1 For automicity.**

**If the case is not maintained -1. ie. Dsh is decrypted to wow as Dsh should be decrypted to Wow.**

**Penalty Matrix:**

| Penalty List | Labs | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 3 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Indentation putting { Infront of loop header, in do while, putting while with closing } | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Meaningful Variable Names |  | -2 | -2 | -2 | -2 | -2 | -2 | -2 |  | -2 | -2 | -2 |  |  |  |  |
| Camel Case Notation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Atomicity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Syntax error | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Linker error | 0 | 0 | 0 | 0 | 0 | 0 | - | - |  | - | - | - |  |  |  |  |
| Wrong function prototypes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Class interface or additional members |  |  | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Use of library function/class without permission | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Continue statement | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| cin/cout where it isn’t needed | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Multi-filing |  |  | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 |  |  |  |  |
| Wrong #ifndef or name of header file |  |  | -2 | -2 | -2 | -2 | -2 | -2 |  | -2 | -2 | -2 |  |  |  |  |
| Global functions |  |  | -3 | -3 | -3 | -3 | -3 | -3 |  | -3 | -3 | -3 |  |  |  |  |
| Multiple classes in one header file |  |  | -3 | -3 | -3 | -3 | -50% | -50% |  | -50% | -50% | -50% |  |  |  |  |