DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

OBJECT ORIENTED PROGRAMMING

Course Code: CT-260

ASSIGNMENT — DESIGN PATTERNS IN REAL-WORLD SOFTWARE SYSTEMS

Semester	Spring 2026	
Class	1st Year — BS Computer Science & Information Technology	
Issued On	Week 7	**Submission Deadline: 30 Days**
Total Marks	5 Marks	Group Assignment, 3 members in a group.
Instructor	Dr. Murk Marvi	
Note	Please prepare this assignment carefully, each of you will have to present the work you have reported in the assignment after mid exam.	

PROJECT :

File encryption suite with multiple ciphers (CipherCascade)

GROUP MEMBERS :

Minahil Khan CT-25066

Rubaisha Arif CT-25067

Task A — Application Overview (1 Marks)

Military encryption is used constantly to protect classified communications, intelligence data, and operational commands from enemy interception. It has been historically critical in conflicts like World War II, where the Allies broke Germany's Enigma cipher and the US used Navajo Code Talkers to transmit unbreakable messages, and during the Cold War when encrypted communications prevented nuclear escalation. In modern warfare, it is used everywhere on the battlefield from encrypted radios and satellite links to drone control systems and nuclear launch authentication, ensuring that only authorized personnel can access sensitive information. The technology works through advanced algorithms like AES-256, hardware-based encryption chips, and secure key distribution protocols, making it virtually impossible for adversaries to decipher intercepted data. Without military encryption, troop movements, airstrike coordinates, and intelligence sources would be exposed, leading to catastrophic operational failures and loss of life.

Task B — Design Pattern Identification (2 Marks)

Pattern 1 — Strategy (Behavioral)

Where does it appears?

In our system, the base class Ciphers defines the common interface:
virtual void encrypt() = 0;
virtual void decrypt() = 0;
The derived classes implement different encryption algorithms such as VigenereCipher , XORCipher , ByteReversalCipher , AtbashCipher.
Each class provides its own implementation of encrypt() and decrypt().

Why Strategy?

If all encryption algorithms were written inside one large class using multiple if-else or switch statements, the code would become difficult to manage and extend. The Strategy pattern allows each encryption algorithm to be implemented in a separate class while sharing a common interface. This makes the algorithms interchangeable and easy to extend without modifying existing code.

OOP principles reinforced

Polymorphism
Abstraction
Open/Closed Principle

Pattern 2 — Template Method (Behavioral)

Where does it appears?

The base class Ciphers handles:
File reading (constructor) , Saving encrypted file (saveEncrypted()) , Saving decrypted file (saveDecrypted()).
The derived classes only implement encrypt() and decrypt().

Why Template Method?
All cipher types follow the same process: read file → apply encryption/decryption → save file. Instead of rewriting file handling code in every cipher class, the base class defines the common structure while subclasses override only the algorithm step. This keeps the workflow consistent and avoids duplication.

OOP principles reinforced
Inheritance
Encapsulation
Code Reusability

Pattern 3 — Chain of Responsibility (Behavioral)

Where it appears?
In the main() function, the program allows the user to select up to three ciphers. After one cipher processes the file, the output file becomes the input for the next selected cipher. The file therefore passes through multiple cipher handlers sequentially.

Why Chain of Responsibility?
Each cipher acts as a handler that processes the file and then passes it forward for further processing. This allows multi-layer encryption without tightly coupling the cipher classes together. The user can decide the order of execution, creating a flexible encryption chain.

OOP principles reinforced
Modularity (dividing a program into modules(separate parts))
Separation of Responsibilities (each class /module should have only one main job)
Loose Coupling (classes are not directly dependent on one another)

Task C — C++ Implementation (1 Marks)

```
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <cstdlib>
#include <algorithm>
using namespace std;

// Base class
class Ciphers {
private:
   string filename;
protected:
```

```
    vector<char> holder;
    int size;
public:
    virtual void encrypt() = 0;
    virtual void decrypt() = 0;

    void saveEncrypted() {
        ofstream outfile("EncryptedVer.enc", ios::binary);
        outfile.write(holder.data(), holder.size());
        outfile.close();
        cout << "Encrypted file saved as EncryptedVer.enc" << endl << endl;
    }

    void saveDecrypted() {
        ofstream outfile("DecryptedVer", ios::binary);
        outfile.write(holder.data(), holder.size());
        outfile.close();
        cout << "Decrypted file saved as DecryptedVer" << endl << endl;
    }

    Ciphers(string n) : filename(n) { // enter name/path
        ifstream file(n, ios::binary);
        if (!file.is_open()) {
            cout << "Error: File failed to open " << n << endl;
            exit(1);
        }
        file.seekg(0, ios::end);
        size = file.tellg();
        file.seekg(0, ios::beg);
        holder.resize(size);
        if (file.fail()) {
            cout << "Error: Failed to read file " << n << endl;
            exit(1);
        }
        file.read(holder.data(), size);
        file.close();
        cout << "Successfully read " << size << " bytes into memory" << endl;
    }

    string getFilename() const { return filename; }
    int getsize() const { return size; }

    virtual ~Ciphers() {}
};
```

//1st Cipher

```
// Atbash Cipher class
class AtbashCipher : public Ciphers {
public:
    AtbashCipher(string n) : Ciphers(n) {}

    void encrypt() override {
        for (int i = 0; i < holder.size(); i++) {
            holder[i] = 255 - (unsigned char)holder[i];
        }
    }

    void decrypt() override {
        for (int i = 0; i < holder.size(); i++) {
            holder[i] = 255 - (unsigned char)holder[i];
        }
    }
};
```

//2nd Cipher

```
// Byte Reversal Cipher class
class ByteReversalCipher : public Ciphers {
public:
    ByteReversalCipher(string n) : Ciphers(n) {}

    void encrypt() override {
        reverse(holder.begin(), holder.end());
    }

    void decrypt() override {
        reverse(holder.begin(), holder.end()); // Reverse back
    }
};
```

//3rd Cipher

```
// Vigenere Cipher class
class VigenereCipher : public Ciphers {
private:
    string key;

public:
    VigenereCipher(string n, string k) : Ciphers(n) {
        key = k;
    }

    void encrypt() override {
        int keyIndex = 0;   // Key index

        for (int i = 0; i < holder.size(); i++) {
            char c = holder[i];

            if (isalpha(c)) {
                char k = key[keyIndex % key.length()];
                int kValue = toupper(k) - 'A';

                if (isupper(c)) {
                    int tValue = c - 'A';
                    holder[i] = ((tValue + kValue) % 26) + 'A';
                }
                else {  // Lowercase
                    int tValue = c - 'a';
                    holder[i] = ((tValue + kValue) % 26) + 'a';
                }

                keyIndex++;   // Move to next key character
            }
        }
    }

    void decrypt() override {
        int keyIndex = 0;

        for (int i = 0; i < holder.size(); i++) {
            char c = holder[i];

            if (isalpha(c)) {
                char k = key[keyIndex % key.length()];
                int kValue = toupper(k) - 'A';

                if (isupper(c)) {
```

```
                int cValue = c - 'A';
                holder[i] = ((cValue - kValue + 26) % 26) + 'A';
            }
            else {  // Lowercase
                int cValue = c - 'a';
                holder[i] = ((cValue - kValue + 26) % 26) + 'a';
            }

            keyIndex++;   // Move to next key character
        }
    }
  }
};
```

//4th Cipher

```
// XOR Cipher class
class XORCipher : public Ciphers {
private:
  int key;
public:
  XORCipher(string n, int k) : Ciphers(n) {
    key = k;
  }

  void encrypt() override {
    for (int i = 0; i < holder.size(); i++) {
      holder[i] = holder[i] ^ key;
    }
  }

  void decrypt() override {
    for (int i = 0; i < holder.size(); i++) {
      holder[i] = holder[i] ^ key; // XOR reverses itself
    }
  }
};
```

// Main program

```
int main() {
    cout << "\n";
    cout << "=================================\n";
    cout << "    C I P H E R  C A S C A D E \n";
    cout << "=================================\n";
    cout << "  Vigenere   XOR   Byte Rev   \n";
    cout << "            Atbash            \n";
    cout << "         3-Step Chain         \n";
    cout << "=================================\n\n";

    int n = 1;
    int choice;
    string file, key;
    int s;

    cout << "Enter the path/file you want to encrypt/decrypt: " << endl;
    cin >> file;
    cout << "Do you want to encrypt the file or decrypt it?" << endl;
    cout << "1. Encrypt" << endl << "2. Decrypt" << endl;
    cin >> s;

    while (n <= 3) {
        cout << "Select " << n << " cipher:" << endl;
        cout << "1. Vigenere Cipher" << endl;
        cout << "2. XOR Cipher" << endl;
        cout << "3. Byte Reversal Cipher" << endl;
        cout << "4. Atbash Cipher" << endl;
        cin >> choice;

switch (choice)
```

//case 1

```
{
        case 1: {
           cout << "Enter string key: " << endl;
           cin >> key;
           VigenereCipher v1(file, key);
           if (s == 1) {
              cout << "Encrypting file through Vigenere Cipher..." << endl;
              v1.encrypt();
              v1.saveEncrypted();
              file = "EncryptedVer.enc";
           } else {
              cout << "Decrypting file through Vigenere Cipher..." << endl;
              v1.decrypt();
              v1.saveDecrypted();
              file = "DecryptedVer";
           }
           break;
        }
```

//case 2

```
        case 2: {
           int k;
           cout << "Enter integer key: " << endl;
           cin >> k;
           XORCipher x1(file, k);
           if (s == 1) {
              cout << "Encrypting file through XOR Cipher..." << endl;
              x1.encrypt();
              x1.saveEncrypted();
              file = "EncryptedVer.enc";
           } else {
              cout << "Decrypting file through XOR Cipher..." << endl;
              x1.decrypt();
              x1.saveDecrypted();
              file = "DecryptedVer";
           }
           break;
        }
```

//case 3

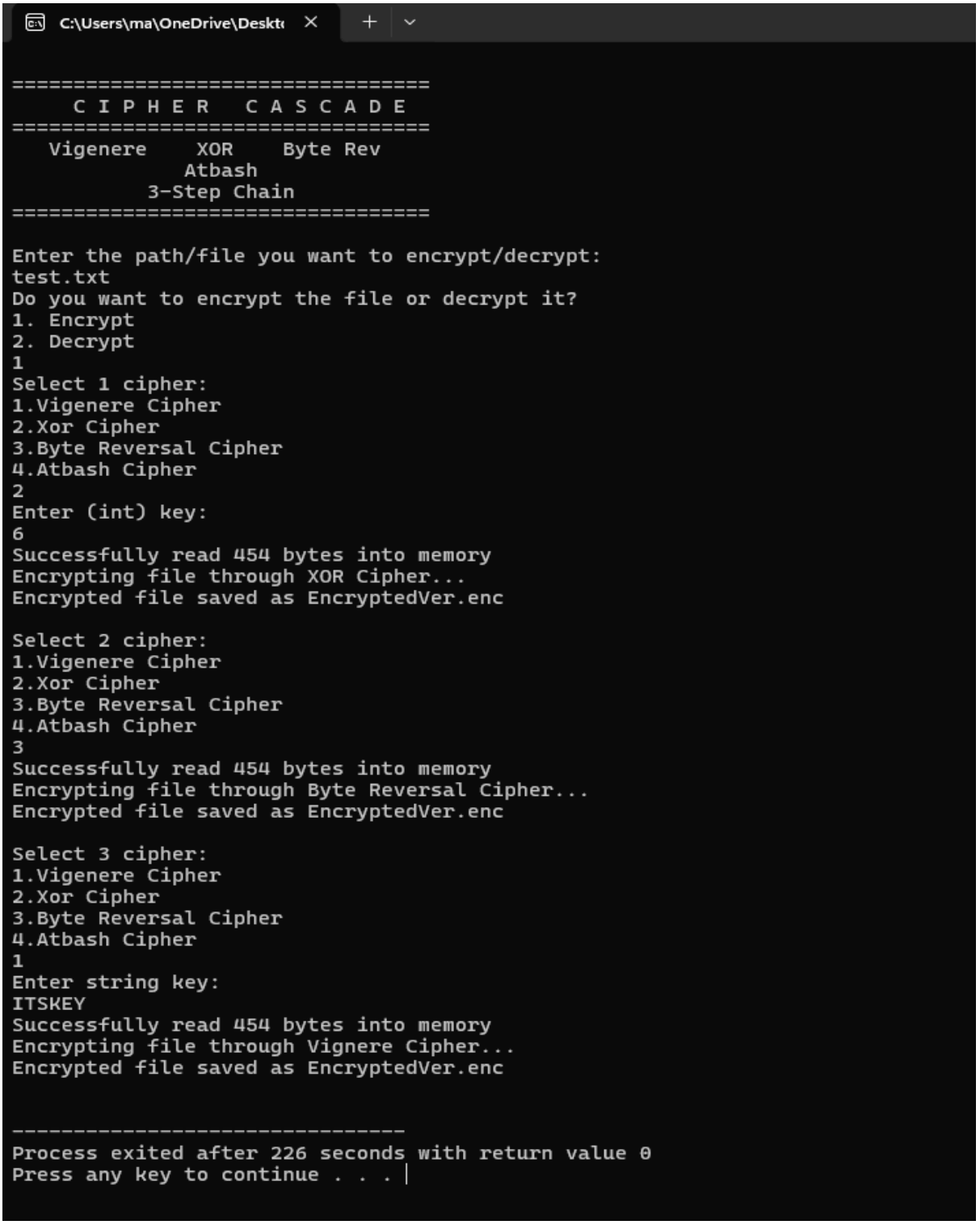
```
        case 3: {
            ByteReversalCipher b1(file);
            if (s == 1) {
                cout << "Encrypting file through Byte Reversal Cipher..." << endl;
                b1.encrypt();
                b1.saveEncrypted();
                file = "EncryptedVer.enc";
            } else {
                cout << "Decrypting file through Byte Reversal Cipher..." << endl;
                b1.decrypt();
                b1.saveDecrypted();
                file = "DecryptedVer";
            }
            break;
        }
```

//case 4

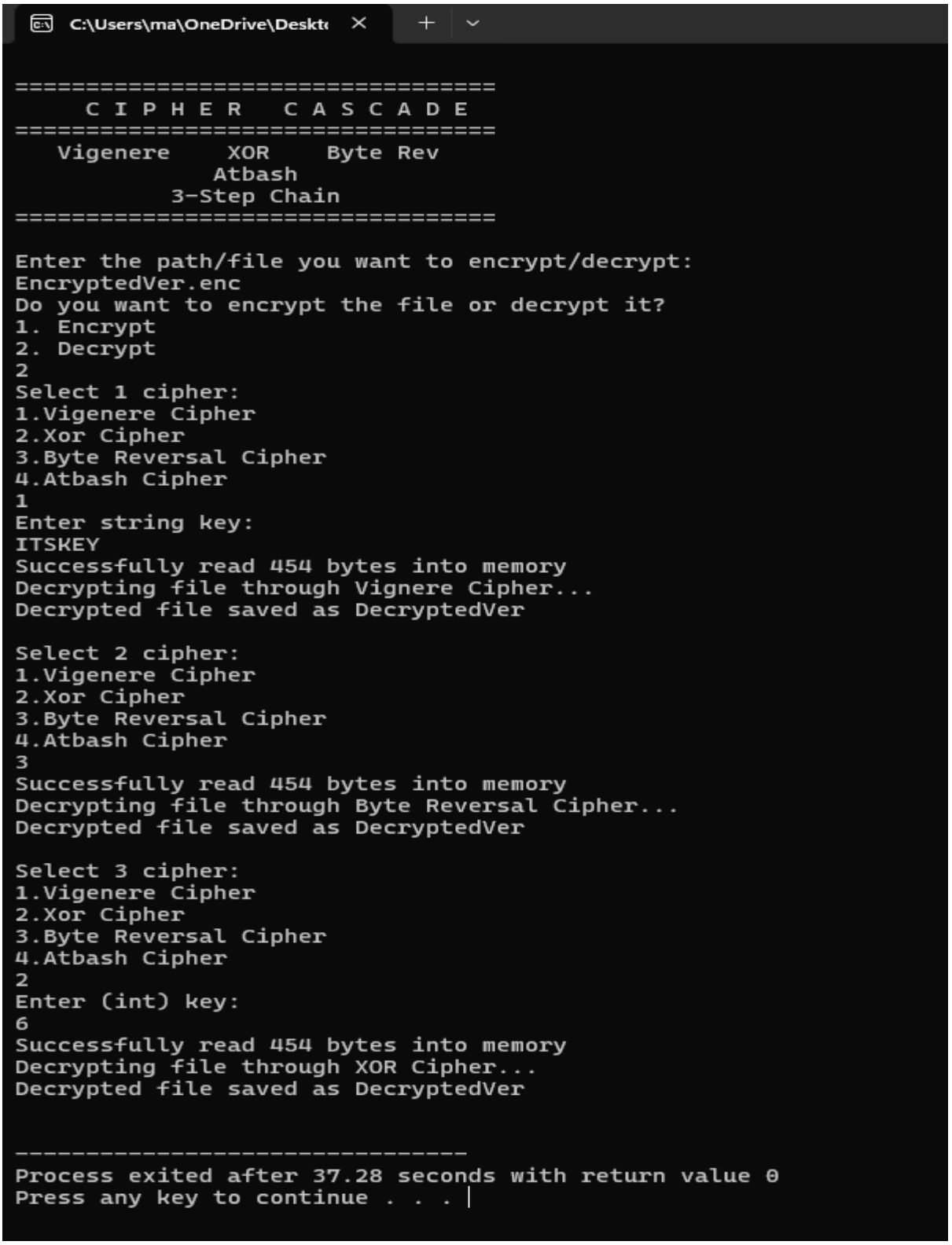
```
        case 4: {
            AtbashCipher a1(file);
            if (s == 1) {
                cout << "Encrypting file through Atbash Cipher..." << endl;
                a1.encrypt();
                a1.saveEncrypted();
                file = "EncryptedVer.enc";
            } else {
                cout << "Decrypting file through Atbash Cipher..." << endl;
                a1.decrypt();
                a1.saveDecrypted();
                file = "DecryptedVer";
            }
            break;
        }
        default:
            cout << "Error: incorrect selection;" << endl;
            break;
    }
    n++;
  }
}
```

OUTPUTS

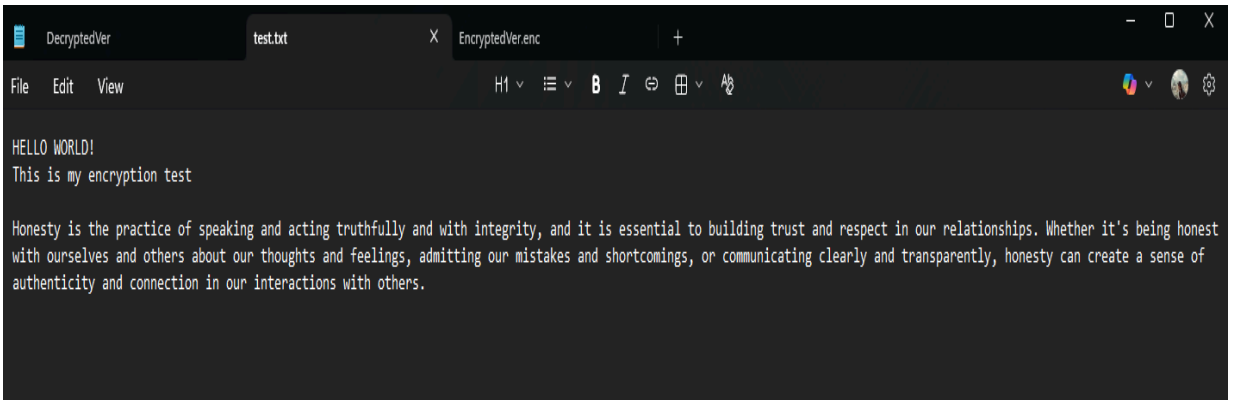
Encryption



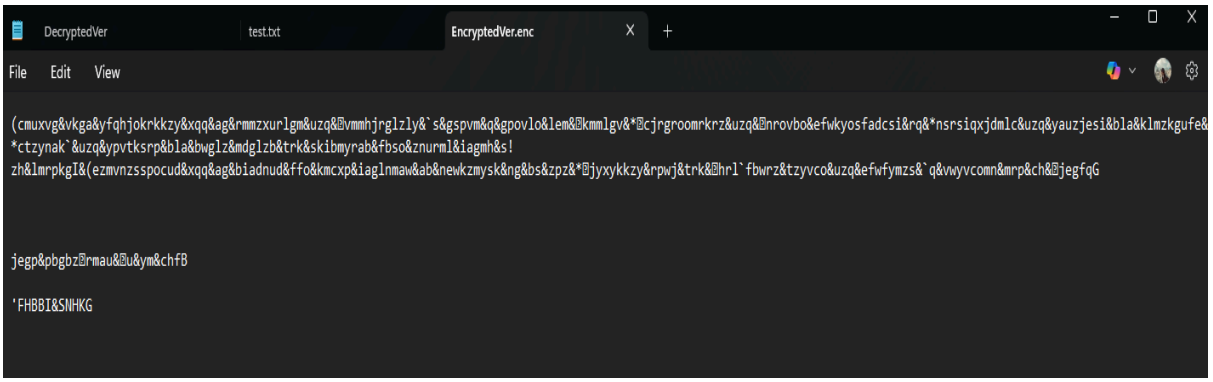
Decryption


C:\Users\ma\OneDrive\Deskt
==========
C I P H E R C A S C A D E
==========
Vigenere XOR Byte Rev
Atbash
3-Step Chain
==========
Enter the path/file you want to encrypt/decrypt:
EncryptedVer.enc
Do you want to encrypt the file or decrypt it?
1. Encrypt
2. Decrypt
2
Select 1 cipher:
1.Vigenere Cipher
2.Xor Cipher
3.Byte Reversal Cipher
4.Atbash Cipher
1
Enter string key:
ITSKEY
Successfully read 454 bytes into memory
Decrypting file through Vignere Cipher...
Decrypted file saved as DecryptedVer
Select 2 cipher:
1.Vigenere Cipher
2.Xor Cipher
3.Byte Reversal Cipher
4.Atbash Cipher
3
Successfully read 454 bytes into memory
Decrypting file through Byte Reversal Cipher...
Decrypted file saved as DecryptedVer
Select 3 cipher:
1.Vigenere Cipher
2.Xor Cipher
3.Byte Reversal Cipher
4.Atbash Cipher
2
Enter (int) key:
6
Successfully read 454 bytes into memory
Decrypting file through XOR Cipher...
Decrypted file saved as DecryptedVer
Process exited after 37.28 seconds with return value 0
Press any key to continue . . .

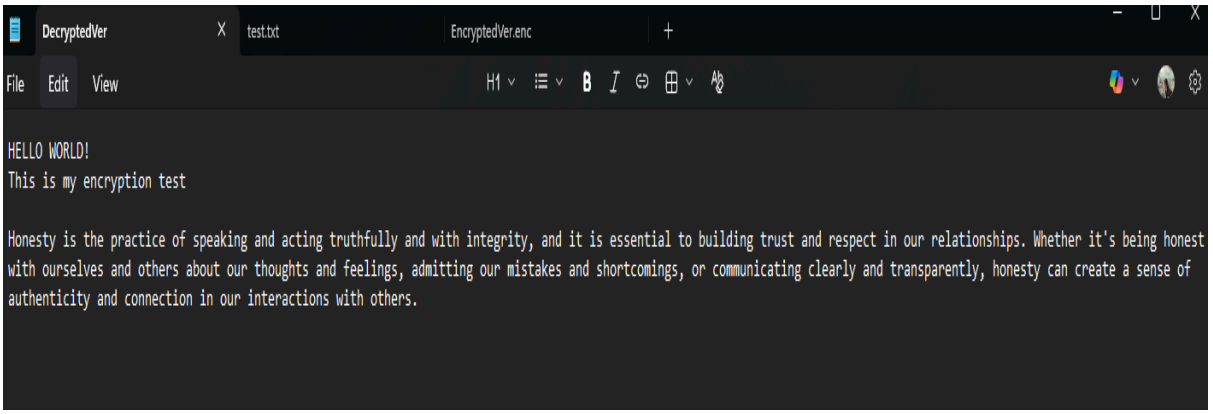
Original Version



Encrypted Version



Decrypted Version



Task D — UML Class Diagrams (1 Marks)