PROJECT DOCUMENTATION

Team name, members: Team Noel, Noel Perez (working alone)

What problems are you solving in this project?

In this project the problems I am trying to solve are successfully encrypting and decrypting a message from a user, using the Vigenere cipher method of polyalphabetic substitution and algorithm. The user should be able to enter any message they wish to have encrypted along with a keyword and the program should encrypt that plaintext and produce a ciphertext. The program should also be able to decrypt a ciphertext back into the plaintext, as long as the user provides the keyword used to encrypt the message initially. The Vigenère cipher can be used to encrypt a message, ensuring that it remains confidential during transmission or storage. It obscures the original content and requires the user to possess the correct key to decrypt and obtain the original plaintext. The Vigenère cipher algorithm protects sensitive, non-critical information, such as personal messages, files, casual correspondence, or other forms of low-risk communication.

What solutions are you implementing in the project?

The solutions I am implementing in this project are the successful encryption and decryption of an uppercase alphabetical character message that contains no spaces or symbols, with the use of a user selected keyword in uppercase characters. The program will also decrypt ciphertexts, as long as the original uppercase plaintext contained no spaces or symbols, and the user provides the correct uppercase keyword used. This program will successfully encrypt information such as passwords, file contents, and personal messages. It provides a simple layer of encryption and decryption that may deter eavesdroppers or unauthorized access to data.

Provide explanation of calculations and algorithm implementation.

The program calculations and algorithm use modular arithmetic to perform the majority of the computations necessary to encrypt and decrypt messages. For example, in the vigenere () function the newKey string is assigned to the value of the cyclically repeated keyword entered by the user, until newKey is the same length as the user's message. If the user's keyword is the same length as their message, then the newKey string would be the same as the keyword they entered. No looping would be required to get the length of the new key equal to that of the plaintext. The vigenere function also accepts a Boolean value which determines which case the user entered from the switch case in main. As far as the algorithm for encryption is concerned, the modular arithmetic formula used is: *newText += (text[index] + newKey[index]) % 26 + 'A'.* The newText string will hold an encrypted character at each index of the string, which together make up the entire ciphertext. The formula for encryption works like this: At each index of each string, the ciphertext is calculated by taking the ascii value of the plaintext letter at that index and adds it to the ascii value of the keyword letter at that same index—this is why the user's message and key must be the same length. The addition of the two values yields one value on which modular 26 is performed. We use 26 because our alphabet only has 26 letters, and the program does not encrypt symbols or spaces. Next the ascii value of 65 or 'A' is added to bring that value into the range of the capital letters ascii values. For decryption the process is similar, but the formula is altered a bit. We use this: newText += (text[index] - newKey[index] + 26) %26+ 'A'.

At each index of the newText string the plaintext is calculated by taking the ascii value of the ciphertext letter at that index and subtracting it from the ascii value of the keyword letter at that same index—the user's ciphertext message and key must also be the same length. The subtraction of the two values yields a value to which we add 26. We add 26 to each index of the decryption process and perform modulus 26 because our alphabet and program only encrypts the 26 alphabetical characters in our alphabet and does not consider spaces or symbols. Next the ascii value of 65 or 'A' is added to bring that value into the range of the capital letter ascii values. This process yields the original plaintext message.

What are the program objectives? Explain how your program is interacting with the user and its purpose.

The program objectives are to successfully encrypt and decrypt a user's message, as long as the plaintext message and keyword are all uppercase letters and contain no spaces or symbols. The program contains a loop that allows the user to continually encrypt and decrypt messages until the sentinel value is entered. The program prompts the user with a menu of 3 options: encrypt, decrypt and exit. The user is free to choose any option. If the user selects encrypt, the program prompts them for the message they would like to encrypt and the keyword they would like to use. After the encryption is done the corresponding ciphertext appears on the screen. The user can then continue the process again or terminate the program. If the user selects decrypt, then the program prompts them for a ciphertext they would like to encrypt. Then it prompts them for the keyword used to generate the ciphertext. This is important because the keyword to decrypt MUST be the same as the keyword used to encrypt or else the decryption process will not yield the correct results. This process goes on until the sentinel value is entered.

How are discrete structures implemented in the C++ program?

Discrete structures are implemented in the C++ program by using logical conjunction statements, "not" logical operator, encoding, modular arithmetic concepts, Boolean logic, and cryptography techniques. The logical statements are used throughout the program in conditional statements and control structures. The switch control structure also uses encoding for each of its cases. Boolean logic is used to produce either encryption or decryption, depending on the user's input. The cryptography techniques used are the Vigenere cipher algorithm which encrypts and decrypts using modular arithmetic along with the encoding ascii system C++ has built in. Since each letter of the alphabet has a numerical value, we can use modular arithmetic to produce encrypted or decrypted text. The ascii encoding system also has numerical values for symbols and other non-alphabetic characters. The vigenere cipher uses a text string and a key for doing polyalphabetic substitution and shifts, which can be visualized with the corresponding Vigenere table.

What are the limitations of the program?

The limitations of this program are that Vigenere cipher is considered relatively weak by modern cryptographic standards. For more secure applications, a stronger encryption algorithm is recommended, such as AES (Advanced Encryption Standard) or RSA (Rivest-Shamir-Adleman). These are generally preferred for secure communication and data protection. Another limitation of this program is that it cannot successfully encrypt or decrypt texts which are not composed of only uppercase alphabetical characters or use keywords that do not also follow these guidelines. This means that a user cannot enter mixed-case texts, put spaces between words, or use punctuation or any other symbols in the program. If the user does include any of the prohibited characters or whitespace the algorithm does not produce the correct results for both encryption and decryption. This has a lot to do with the ascii encoding system and the fact that this encoding system has a lot of scattered elements which are difficult to do modular arithmetic on especially with our equations for encryption and decryption using the Vigenere cipher.

Provide recommendation on improving the limitations of the program.

Some recommendations for improving the limitations of the program might include using a different cryptographic method depending on the sensitivity of our data. Furthermore, this program would need a more robust data structure approach or container for successfully encrypting mixed case texts and symbols. The C+ STL has a table container that could be very useful for improving this program, which could help solve the encoding system problem. I think to successfully encrypt and decrypt any type of texts, more than discrete structures methods are needed. The issue with the encoding system is that there are different bounds for uppercase and lowercase letters which need to be resolved. Also, the symbols are scattered randomly in an unsystematic way making it difficult to do modular arithmetic on them. This is why a table with the entire alphabet in upper and lowercase, along with all the necessary symbols could be created and assigned a numerical value which would make doing the necessary modular arithmetic in a systematic way easier. Also, data validation could be used to ensure only valid data is entered.

PSEUDOCODE

```
INCLUDE iostream
INCLUDE string
USING namespace std;
Program beginning
MAIN FUNCTION START
        Declare char choice
        Declare bool decode
        Declare string plainText
        Declare string keyWord
        Declare string cipherText
        Declare char c
        DISPLAY: “PROGRAM ONLY ACCEPTS UPPERCASE ALPHABETICAL
        CHARACTERS (A-Z): "
        DISPLAY: "NO SPACES OR SYMBOLS."
        DISPLAY: “PLEASE ONLY ENTER UPPERCASE MESSAGES AND KEYWORD
        WITHOUT SPACES OR SYMBOLS!
 DO LOOP START
     Declare int choice.
     DISPLAY: "Choose from the menu: "
     DISPLAY: "1: Encrypt"
     DISPLAY: "2: Decrypt"
     DISPLAY: "3: TERMINATE PROGRAM "
     INPUT choice
        SWITCH CASE START (Read choice)
            CASE 1: Encrypt plaintext message into ciphertext
                decode=0
                DISPLAY: "Enter message to encrypt:”
                INPUT plainText
                DISPLAY: "Enter keyword: "
                INPUT keyWord
```

```
                 // CALL VIGENERE FUNCTION (Pass in plainText, keyWord, decode)
                cipherText = vigenere () // FUNCTION CALL
                DISPLAY: "The ciphertext is: "
                OUTPUT: cipherText
                DISPLAY: "Continue Y/N? "
                 INPUT c
                 c = toupper (Read c)
                 BREAK; END OF CASE 1
            CASE 2: Decrypt ciphertext into plaintext
                decode = 1
                DISPLAY: "Enter ciphertext to decrypt: "
                INPUT cipherText
                DISPLAY: "Enter keyword used: "
                INPUT keyWord
                // CALL VIGENERE FUNCTION (Pass in cipherText, keyWord, decode)
                plainText= vigenere () // Function call
                DISPLAY: “The plaintext is: "
                OUTPUT plainText
                DISPLAY: "Continue Y/N?"
                INPUT c
                c = toupper (Read c)
                BREAK; END OF CASE 2
            CASE 3: Exit program
                DISPLAY: "EXITING PROGRAM! THANK YOU!"
                 c = 'N'
                BREAK; END OF CASE 3
      END OF SWITCH CASE
    WHILE (c is equal to 'Y' AND choice does not equal 3)
    END OF DO-WHILE LOOP
return 0
```

```
END OF MAIN FUNCTION

START OF VIGENERE FUNCTION DEFINITION

FUNCTION string vigenere (Parameters are string text, string key, bool encode)

   DECLARE string newKey

   DECLARE string newText

   newKey = key

   START OF WHILE LOOP

   WHILE (The length of newKey is less than the length of text)

         newKey += key   /* newKey should be assigned the value of the keyword repeated

                          * cyclically until it is the same length as the user's message text.  */

   END OF WHILE LOOP

      IF the value of decode is equal to 0

         FOR (int index=0; index is less than the length of text; increase index)

           newText += the ascii value of the letter in the plaintext [index] plus (+) the ascii value
           of the newKey[index] modulus 26 and add 65 to bring it in range of the capital letter
           ascii values.

            // newText += (text[index] + newKey[index]) % 26 + 'A'

         ENDFOR

      ENDIF

     ELSE

          FOR (int index =0; index is less than the length of the text; increase index)

           newText += the ascii value of the letter in the plaintext [index] minus (-) the ascii value
           of the newKey[index] plus 26 then modulus 26 of that value. Now add 65 to bring it in
           range of the capital letter ascii values.

            // newText += (text[index] - newKey[index] + 26) %26+ 'A'

          ENDFOR

      END ELSE

   return newText

END OF VIGENERE FUNCTION

End of Program
```