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 same length as original key

**END OF WHILE LOOP**

**IF** the value of decode is equal 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**

/\* For encryption, newText is computed by adding the value of each index from the original text and the keyword index and doing modular arithmetic on those values; after that has been done the ascii value of ‘A’ which is 65 is added to bring it in range of the capital letter ascii value. For decryption, the index values are subtracted and 26 is added to get the inverse of the modular arithmetic performed in the encryption process. ‘A’ or 65 is added to bring it in range of the capital letters ascii values and the result yields the original message \*/