Pseudo Code

**Main:**

BEGIN PROGRAM

Acquire input from User

FUNCTION runSpellChecker

SET file TO "Dictionary.txt"

CREATE empty list: dictionaryList

CREATE empty list: userWords

CREATE empty string: wordsMisSpelled

CREATE SpellChecker object: sp

dictionaryList ← sp.GetDictionaryAsVector(file)

SET file TO "Words.txt"

userWords ← sp.GetUserWordList(file)

wordsMisSpelled ← sp.WordsNotInDictionary(dictionaryList, userWords)

DISPLAY "The following words are in the users word list:"

FOR EACH word IN userWords

DISPLAY word

DISPLAY "The following words are not in the systems dictionary:"

DISPLAY wordsMisSpelled

END FUNCTION

FUNCTION runDecrypter

CREATE DecrypterClass object: dc

SET file TO "Message.txt"

DECLARE keyword, message, encryptedAlphabet, encryptedMessage, decryptedMessage

DECLARE EncryptOrDecrypt

REPEAT

PROMPT "Enter a keyword for cypher"

READ keyword

UNTIL keyword IS NOT empty

encryptedAlphabet ← dc.CreateCypher(keyword)

PROMPT user: (1) Encrypt or (2) Decrypt

READ EncryptOrDecrypt

IF EncryptOrDecrypt IS 1 THEN

PROMPT user to enter a message

READ message (entire line)

encryptedMessage ← dc.EncryptMessage(message, encryptedAlphabet)

dc.WriteMessageToFile(file, encryptedMessage)

ELSE IF EncryptOrDecrypt IS 2 THEN

encryptedMessage ← dc.GetMessageToDecrypt(file)

decryptedMessage ← dc.DecryptMessage(encryptedMessage, encryptedAlphabet)

DISPLAY decryptedMessage

ELSE

DISPLAY "Invalid selection"

END FUNCTION

FUNCTION main

PROMPT user to choose a program: 1 for spell checker, 2 for decrypter

READ choice

IF choice IS 1 THEN

CALL runSpellChecker

ELSE IF choice IS 2 THEN

CALL runDecrypter

ELSE

DISPLAY "invalid entry"

END FUNCTION

END PROGRAM

**Spell Check:**

METHOD Constructor:

Initialize any internal variables if needed

METHOD Destructor:

Perform cleanup if needed

FUNCTION GetUserWordList(filepath):

CREATE empty list userWords

TRY to open file at filepath

IF file cannot be opened THEN

DISPLAY error message

RETURN empty userWords list

WHILE there is data to read from the file

READ word

CONVERT word to uppercase

ADD word to userWords list

CLOSE the file

RETURN userWords

FUNCTION GetDictionaryAsVector(filepath):

CREATE empty list wordlist

TRY to open file at filepath

IF file cannot be opened THEN

DISPLAY error message

RETURN empty wordlist

WHILE there is data to read from the file

READ word

CONVERT word to uppercase

ADD word to wordlist

CLOSE the file

RETURN wordlist

FUNCTION WordsNotInDictionary(DictionaryList, UserWords):

CREATE empty string wordsNotInDictionary

FOR EACH word IN UserWords:

IF word is NOT in DictionaryList THEN

APPEND word and a space to wordsNotInDictionary

RETURN wordsNotInDictionary

**Decrypter:**

CLASS DecrypterClass:

METHOD Constructor:

Initialize any necessary internal variables

METHOD Destructor:

Perform cleanup if required

FUNCTION WriteMessageToFile(filename, message):

OPEN output file with given filename

IF file is successfully opened THEN

WRITE message to file

CLOSE file

DISPLAY success message

ELSE

DISPLAY error message

ENDIF

FUNCTION CreateCypher(keyword):

SET baseAlphabet to "a" through "z"

INITIALIZE empty string processedKeyword

INITIALIZE empty string remainingAlphabet

INITIALIZE seenInKeyword array[26] as false

FOR each character c in keyword:

CONVERT c to lowercase

IF c is a letter AND not already seen in keyword THEN

ADD c to processedKeyword

MARK c as seen

ENDIF

ENDFOR

INITIALIZE seenInAlphabet array[26] as false

FOR each character c in processedKeyword:

MARK c as seen in seenInAlphabet

ENDFOR

FOR each character c in baseAlphabet:

IF c not seen in seenInAlphabet THEN

ADD c to remainingAlphabet

ENDIF

ENDFOR

REVERSE remainingAlphabet

RETURN processedKeyword + remainingAlphabet

FUNCTION EncryptMessage(message, encryptedAlphabet):

SET standardAlphabet to "a" through "z"

INITIALIZE empty string encryptedMessage

FOR each character c in message:

IF c is a letter THEN

CONVERT c to lowercase

FIND index of c in standardAlphabet

APPEND character at same index from encryptedAlphabet to encryptedMessage

ELSE

APPEND c to encryptedMessage

ENDIF

ENDFOR

RETURN encryptedMessage

FUNCTION GetMessageToDeCrypt(filepath):

OPEN input file at filepath

INITIALIZE empty string message

IF file cannot be opened THEN

DISPLAY error message

RETURN empty message

ENDIF

WHILE reading words from file:

CONVERT word to lowercase

APPEND word to message (no spaces)

ENDWHILE

CLOSE file

RETURN message

FUNCTION DecryptMessage(message, encryptedAlphabet):

SET standardAlphabet to "a" through "z"

INITIALIZE empty string decryptedMessage

FOR each character c in message:

IF c is a letter THEN

CONVERT c to lowercase

FIND index of c in encryptedAlphabet

APPEND corresponding standard letter plus a space to decryptedMessage

ELSE

APPEND c plus a space to decryptedMessage

ENDIF

ENDFOR

RETURN decryptedMessage