Report

A.

-Mapping the key and having the values translate to uppercase chars when decrypting

-Learning how to use a 2-D array to filter out the spaces and non alphas

-Checking and filtering multiple spaces and non-alphas

B.

The first helper function, getCribLength, counts all the alphas in the crib to return the number of words it has.

The second helper function, maxlength is to get the max length of the crib – used only to see if the crib is above 80 chars or 0 chars, both in which can’t be used to translate the ciphertext.

The third helper function, charToLower, loops through a char array and lowercases all the alphas

The fourth helper function, isTranslatable, checks my key to see if it’s a valid key to use for translating. It loops through the key using two idices, with one index being one more than the other in order to check for duplicate values to one key.

The fifth helper function, translation, utilizes the key we made

For the decrypt() function:

If the length of the crib is greater than 80 or 0, return false

Create a new char array the length of the crib

Copy the crib into the new char array

Lowercase all the characters

Create a 2-D array with the number of rows matching the crib length and with 81 columns as its max (80 characters including the ‘\0’)

Loop through the copied crib

If the loop encounters a letter in the crib, store it in the corresponding row

and keep storing for each column while there are consecutive letters

After storing, add a zero byte to make this a word

If the loop encounters anything but an alpha, move on the the next row while incrementing until we encounter an alpha again

When we do encounter an alpha, decrement so we start on the alpha instead of skipping over it

Create a new char array the length of the ciphertext

Copy the ciphertext into the new char array

Lowercase all the characters

Create a 2-D array the maximum possible size the ciphertext can be

Initialize a counter for the number for storing the ciphertext’s SENTENCES(all the words before a newline)

Initialize an index to loop through the ciphertext copy with

While looping through the ciphertext copy,

If we encounter a letter,

Initialize an integer to move through the 2-D arrays columns,

Store the consecutive chars into a row column by column until the index is not at an alpha.

If we encounter new line marker,

Move onto the next row

Start on the first column

Store a zero byte in the first column to make sure we’re storing a new sentence

Continue iterating through the ciphertext copy

If we encounter anything but an alpha or newline marker,

Move onto the next row and stop at the first letter so we can begin to store another sentence

TO COMPARE

Initialize a key of 2 columns and the maximum size a the ciphertext can be

Initialize a counter for the crib char in the key

Initialize a counter for the text char in the key

Initialize a counter for the key’s length

Initialize an index p,

While p is less than the number of words in the ciphertext ,

If the first word in the crib matches a word in the text go into the rest of the conditions,

If not, and the text doesn’t have enough words to match the length of the crib, the crib is invalid and return false – we can’t translate this text

If there was a match,

Initialize an index to move through the rest of the crib

Initialize a counter to see if we can match the entire crib

While the words in our 2-D cipher array matches with the words in our 2-D crib array,

Increment the counter

Increment both indexes

If we find a match, but there is a duplicate word, meaning we’ve counted more matches than we should have for our crib,

Subtract the index by the number of matches we’ve made so we can skip over duplicate matches

If the number of matches matches the number of words in our crib, generate the key

Initialize index, x to determine how many words to copy

While index x is less than the number of crib words,

Initialize y to get each char of the crib array’s words,

While y is less than the length of the crib in the respective row,

Store the letter of the crib in first column(0) of key

Increment cribkey to move to the next row for the next letter

Increment y to get the next letter for the next row

Increment x after we copied a word’s letters into their respective rows

Initialize index z to start at the same position as the first letter of the crib in the key,

While z is less than the number of loops it took to match the crib,

Initialize y to move through the text array,

Store the matched text letter in the second column(1) that in the row corresponding to textkey

Increment for the next letter

Increment for the next row

Move onto the next word

If the number of crib letters in the key matches the number of text letters in the key,

Make keylength equal to this number

Use isTranslatable to check if the key is valid,

If it is, use it in the translation function to translate to decrypt the ciphertext

Print out the translation

Return TRUE

If the key isn’t valid, there might be another match later on in the ciphertext,

Restart your key to prepare to make it again

Continue iterating through the text

If everything before this fails, we could not properly match the ciphertext and crib to make a key, so return FALSE

c.

//**Given test cases**

//runtest("Hirdd ejsy zu drvtry od.\nO'z fodvtrry.\n", "my secret"); //runtest("Hirdd ejsy zu drvtry od.\nO'z fodvtrry.\n", "shadow");

**//Tests for puctuations**

//runtest("Hirdd ejsy dr d^^^645847r dr dr dr vtry od.\nO'z fodvtrry.\n", "ni haip");

**//Tests for duplicate matches**

//runtest("Hirdd ejsy dr dr dr dr dr vtry od.\nO'z fodvtrry.\n", "ni haip");

//runtest("Hirdd hi hey hi hey hi halo od.\nO'z fodvtrry.\n", "pi pey pi palo");

//runtest("am kzm hi hey hi halo od.\nO'z fodvtrry.\n", "pi pey pi palo");

**//Tests for a newline separated match**

//runtest("bwra wmwt\n qeirtk spst\n", "alan turing");

//runtest("bwra wmwt QEirtk spSt\n", "alan tuRIng");

**//One char crib test**

//runtest("Hirdd ejsy dr d dr dr dr vtry od.\nO'z fodvtrry.\n", "a");

**//Good match**

//runtest("fodvtr Hirdd ejsy dr d^^^645847r dr dr dr vtry od.\nO'z .\n", "hzrdd");

**//Tests for a wrong key**

//runtest("Hirdd ejsy dr d^^^645847r dr dr dr vtry od.\nO'z ftdvtr.\n", "nihalp");

**//Tests for a match later on in the ciphertext after a potential match**

//runtest("Hirdd ejsy ftdvtr dr d^^^645847r dr dr dr vtry od.\nO'z ftdvir.\n", "nihalp"); /

**//Tests a ciphertext with a longer key**

//runtest("Hirdd ejsy ftdvtr dr d^^^645847r dr dr dr vtry od.\nO'z ftdvir.\n", " orihh wpsy tfhqfi hi");

**//crib words longer than text**

//runtest(" ftdvtr.\n", "nihaip to");

(Returns false)

/**/Same chars in crib**

//runtest("rrrrrr rr rrr r.\nO'z fodvtr.\n", " pp ppp p");

**//Empty key**

//runtest("rrrrrr rr rrr r.\nO'z fodvtr.\n", "");

(Returns false)