1.

A file named **report.docx** or **report.doc** (in Microsoft Word format), or **report.txt** (an ordinary text file) that contains:

The major obstacles were working with two-dimensional arrays and keeping the indexing straight. Also, determining how to keep a record of the location of newline characters was tricky.

**Program design:**

2.

The decrypt function calls all other functions. This program outline represents, at a high level, what the decrypt function is doing.

Determine lengths of crib and cipher

Create duplicates of the crib and cipher that can be manipulated

Call **countWords** to determine how many possible words are in the cipher and crib

**countWords** returns the number of consecutive alphabetic character sequences in the crib and cipher

If the crib is empty, return false

If the crib is longer than the ciphertext, return false

If the crib is longer than 80 characters, return false

If there are more words in the crib than in the ciphertext, return false

If none of the characters in the ciphertext are alphabetic, return false

Assuming all the preliminary tests come back good, decrypt then executes the following set of procedures:

1.) call **createWordArray** to create 2 two-dimensional char arrays: one for the ciphertext, one for the cribtext. Each array represents the words in ciphertext and cribtext

2.) call **wordOrder** to create 2 one-dimensional int arrays that capture -- IN ORDER -- the word lengths in the two-dimensional array

3.) Call the **findMatch** function, that searches for a possible matching sequence in the int array. The implementation of the **findMatch** function calls two test functions -- **containsNewLine** and **isSimpleSub** -- which are detailed below. If a match is found, and it passes the test functions, **findMatch** returns an int that represents the row of two-dimensional char array representing the words of the cipher where the matching sequence begins. The **findMatch** function returns a value of -1 if no potential matches are found; **decrypt** returns false if **findMatch** returns -1.

a) A crib cannot match a sequence of the cipher that contains a newline character; the **containsNewLine** function returns true if there is such newline (represented by '\*') that is not at the start or end of the matching sequence. If this function returns true, the sequence cannot be matched to the crib.

b) The substitution scheme must be one-to-one, e.g. if every E is replaced by K, no other letter will also be replaced by K; the **isSimpleSub** function returns true is the substitution scheme is one-to-one. If this function returns true, the sequence may be matched to the crib, provided it does not contain a newline character.

4.) Assuming there is a good match, i.e. a sequence of the ciphertext that has the right words lengths in the right order, call **justLetters** for the crib and the matching sequence of the cipher. **justLetters** returns a one-dimensional array of alphabetic characters. When called on the matching sequence of the cipher and the crib, it places their letters in corresponding indices.

5.) Call **exchangeLetters** using the 2 one-dimensional arrays created by **justLetters**. **exchangeLetters** searches the duplicate ciphertext for letters matching any letters in the matching sequence of the cipher. If a letter matches a letter in that sequence, **exhangeLetters** replaces that letter in the duplicate ciphertext with the corresponding letter in the array created by **justLetters** for the crib.

5.) Print out the partially-decrypted duplicate cipherText.

6.) return true

**Test Cases**

3.

**char** testcipher1[] = "Vlghjl\n.hj oiup mgak zfefyfekt kyy.\nIfk!\n cddceb cd ncpq";

**char** testcipher2[] = “asssss fd fags”;

**char** testcipher3[] = “4324323&)\*”

**char** testcipher4[] = “ “

**char** testcipher5[] = “fdsss fd”

**char** testcipher6[] =

**char** testcriba[] = "Attack at dawn!";

**char** testcribb[] = "";

**char** testcribc[] = "tank";

**char** testcribd[] = “The purpose of this paper is to consider a gloomy possibility raised by W. K. Frankena in the closing pages of “The Naturalistic Fallacy.” Frankena argues convincingly that the real point of contention between intuitionists and what he calls definists”

(assert(testcipher1, criba) == true) This is a pretty major test that satisfies many corner cases: (1) The first possible matching sequence has to be rejected because it contains a newline character in the middle. (2) A second sequence has to be identified. (3) The crib is multiple words long. (4) There is a newline character in between two non-alphabetic characters.

(assert(testcipher1, cribb) == false) The crib is empty

(assert(testcipher2, criba) == false) The substitution scheme would not be simple

(assert(testcipher4, criba) == false) The crib is longer than the test cipher, which will always be the case if the cipher is zero and the crib cannot be zero.

(assert(testcipher3, cribc) == false) Cipher contains only non-alphabetic characters

(assert(testcipher2, cribc) == false) Cipher letter replaced by itself in simple substitution

(assert(testcipher5, criba) == false) More words in crib ttan in cipher; also, crib is simply longer than cipher

(assert(testcipher1, cribd) == false) crib is more than 80 letters long