1. **Notable Obstacles**

One notable obstacle that I encountered was figuring out how I was going to iterate through ciphertext in sections that were as long as the length of the crib. I wanted my program to only look at sections of ciphertext so that it can determine whether that section corresponds with the crib. After many hours of deliberation, I decided I would create 2 different for loops. In the first for loop, I would have it iterate through all of copytext, which is ciphertext with all of the unnecessary characters taken out, and in the second for loop, I would have the program count different kinds of “matches” (matches of characters, matches of spaces, matches of corresponding letters, etc.) with the restriction of the length of crib to check each section. In addition, in the second for loop, I had the initial statement start at “i” so that the loop would begin at the start of the section in interest. I would then continue the second loop until it iterates the length of crib through copytext or the section in interest hits the end of copytext. Overall, this allowed me to look at incremented sections of copytext and it was able to determine where in copytext crib would be able to match. Another notable obstacle that I encountered was differentiating processes when crib had no spaces to when crib had at least one space. Towards the latter of the project, I ran into a problem where the program would only correctly output the ciphertext when there was at least one space in crib. Most times when crib was just one word and there were no spaces in crib, it wouldn’t correctly output the ciphertext decryption. I fixed this problem by creating an “if-else” statement where the statement would split into if “there are no spaces in crib” and else “when there is at least one space in crib”. With this separation, I was able to split this issue and by creating a separate code section for the “no space in crib” part, I was able to accurately output the decrypted ciphertext for both scenarios.

1. **Pseudocode**

* Corresponding highlights represent matching “if-else” statements

bool decrypt(const char ciphertext[], const char crib[]) {

copy crib into copycrib

change all non-alphabet characters into spaces and make all copycrib lowercase

delete all irrelevant spaces in copycrib

if length of copycrib is greater than 90 return false

if crib string has no words return false

copy ciphertext into plaintext

make all letters in plaintext lowercase

copy plaintext into copytext

change all punctuation characters in copytext into spaces

delete spaces when there are at least two spaces in a row within copytext

if length of copycrib is greater than length of copytext return false

count number of spaces/punctuation characters in crib

count number of spaces in copycrib

count number of matches in copycrib

count number of alphabet characters in copycrib

repeatedly iterating through all of copytext:

count corresponding matches for both copycrib and copytext

count number of digits in sectioned portion of copytext

count number of newline characters in portion of copytext

count number of matches in copytext

count number of times copycrib and plaintext have same kind of character

count number of alphabet characters in very first section

count number of spaces/punctuation characters in section of copytext

if there are no spaces in copycrib

if crib matches first word in copytext

if character after crib length in copytext isn’t a letter

if correlating positions match for copytext and copycrib

if each position in copytext and copycrib have same kind of character

if no numbers and no newline characters

if number of alphabet characters at front is equal to section length

match plaintext with section of copytext

substitute chars across plaintext and capitalize each changed letter

output plaintext

return true

otherwise, crib matches during middle/end of copytext

if iteration gets to last character in copytext

if correlating positions match for copytext and copycrib

if each position in copytext and copycrib have same kind of character

if no numbers and no newline characters

if copycrib is a single letter

if char to the left of ‘i’ is not an alphabet character

match plaintext with section of copytext

substitute chars across plaintext and capitalize each changed letter

output plaintext

return true

return false

otherwise, when ‘i’ is not at last character and iterates through copytext

check that char before section and after section are not alphabet chars

if correlating positions match for copytext and copycrib

if each position in copytext and copycrib have same kind of character

if no numbers and no newline characters

match plaintext with section of copytext

substitute chars across plaintext and capitalize each changed letter

output plaintext

return true

else there is at least one space in copycrib

if number of spaces in section of copytext match with spaces in crib

if correlating positions match for copytext and copycrib

if there are no numbers and no newline characters

if each position in copytext and copycrib have same kind of character

if ‘i’ is equal to zero

if next position after section in copytext is not a letter

match plaintext with section of copytext

substitute chars across plaintext and capitalize each changed letter

output plaintext

return true

otherwise when ‘i’ is not at zero

while next position after section in copytext is not a letter

match plaintext with section of copytext

substitute chars across plaintext and capitalize each changed letter

output plaintext

return true

return false

}

1. **Test Cases**

* decrypt("hey12 jaden@@ lee", "joy"): tests when there are numbers and punctuation characters in ciphertext
* decrypt("a a c da", "j"): tests when the crib is a single letter and crib matches the word at the front of ciphertext
* decrypt("F lgr rntoy rkwndyk ahna'y\nphklk ahk mgtks fy.\n", "J"): tests when there is a newline character in ciphertext
* decrypt("adb 123 db", "jas"): tests when there is no space in crib and the match is in the front of ciphertext
* decrypt("ew'q p-aj", "he"): tests when there are multiple possible matches for crib in ciphertext
* decrypt("happy12 birthday@ hah! you", "jkl"): tests when there is no space in crib and the match is later on in ciphertext
* decrypt("Hirdd ejsy zu drvtry od.\nO'z fodvtrry.\n", "shadow"): tests a false case
* decrypt("12 33 221 342", "ja"): tests when there are only numbers and spaces in ciphertext
* decrypt("12 33 221 342 1", "j"): tests when there are only numbers and spaces in ciphertext and crib is only one character
* decrypt("12aj jk3# #@sa j", "Z"): tests when crib is only a letter and the match is at the end of ciphertext
* decrypt("jesus loves me this i know", "bub"): tests a false case
* decrypt("jesus@@ loves me123 this i know", "ja"): tests when the match has numbers following it
* decrypt("", "ja"): tests when ciphertext is an empty string
* decrypt("Xbg'j rsj jds jsrrsy jycn jds UcRSgJ qrqyt.", "sILent aLarm"): tests when crib has a space in it
* decrypt("Hirdd ejsy zu drvtry od.\nO'z fodvtrry.\n", "my secret"): tests when crib has at least one space and there is a newline character in ciphertext
* decrypt("abcd efg hijk ilm nppq rst dccb cba uwwx wyz", "good one"): tests very specific matches when numerous sections could seem to be matches to crib
* decrypt("wmwt\nqeirtk spst\n", "alan turing"): tests when there is a newline character in the middle of a match
* decrypt("Kpio't dmpbl-boe-ebhhfs opwfm", "s cloak and"): tests when a crib ranges over multiple words in ciphertext
* decrypt("F gspt fe! zyxZYXzyx--Abca abCa bdefg## $$dsptrqtj6437 wvuWVUwvu\n\n8 9\n", "hush hush 1232????111until november?? \n?"): tests when there are punctuation characters and numbers in both ciphertext and crib
* decrypt("I love jesus yes i do", "abdefeg"): tests when crib length is longer than any possible word in ciphertext
* decrypt("jaden lee", "jaden leeyt"): tests when crib is longer than ciphertext
* decrypt("jaden lee", "jaden @#123lee"): tests when there are punctuation characters and numbers in crib
* decrypt("jaden lee", "jad@#123en lee"): tests when numbers and punctuation characters are in between letters of crib