

**ADVDISC**

Machine Project 2 Documentation

|  |  |
| --- | --- |
| **Section** | S19 |
| **Team Members** | Amadora, Angelo John  Andres, John Joseph  Fernandez, Ryan Austin  Syfu, Jonah Espiritu |
| **Date Submitted** | December 1, 2015 |

**Table of Contents**

I. Contribution of Each Member / 3

II. Introduction of the Research Area / 4

III. Use of Matrices and Linear Algebra Methods in the Research Area / 5

IV. Design / 9

V. Implementation / 11

VI. Conclusion / 12

VII. References / 13

Appendix A – Test Cases / 14

**I. Contribution of Each Member**

|  |  |  |
| --- | --- | --- |
| **Member** | **Code** | **Other** |
| Andres, John Joseph | ModArith | Testing |
| Amadora, Angelo John | CipherPanel  DecipherPanel  MainPanel  MainWindow | Quality Assurance |
| Fernandez, Ryan Austin | AbstractMatrix  ModularMatrix  Cipher  CofactorExpansion  FileManager  ModArith  Matrix  CipherController  CipherPanel  DecipherPanel  DerivePanel  MainPanel  CipherInputFrame  MainWindow  Driver  AGBLayout  InvalidCipherException  IController | Documentation  Research |
| Syfu, Jonah Espiritu | ModArith  Cipher | Testing |

**II. Introduction**

Linear Algebra is a very useful field of mathematics, especially in the field of computer science, particularly the concept of linear transformations. Linear transformations are used liberally in the field of cryptography.

Cryptography is the study of techniques in securing messages or text by enciphering them using a set algorithm. The original text is referred to as “plaintext” and the enciphered text is referred to a “ciphertext”. Decoding the ciphertext is referred to a “deciphering”. Finding the algorithm used is referred to as “cracking” or “breaking” a cipher.

The algorithm this project aims to implement is the Hill Cipher algorithm. The objective of the project is to implement the enciphering, deciphering, and the breaking of the Hill Cipher.

**III. Use of Matrices and Linear Algebra Methods in the Research Area**

**Enciphering Using the Hill Cipher**

A Hill-n Cipher makes use of an n x n matrix A and a modulus n. The only requirements of the matrix are that its determinant has a modular inverse with respect to the modulus. This means it does not have any common prime factors.

Once an appropriate matrix is found, a plaintext string is enciphered by first transforming it into a matrix. This is done by first representing the string as its numerical representation.

Assume the scheme is the number’s cardinality in the English Alphabet. Take the string BEATRICE. Our modulus is therefore 27 since the values 0 – 26 are being used, 0 representing a whitespace. This would be represented as

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B | E | A | T | R | I | C | E |
| 2 | 5 | 1 | 20 | 18 | 9 | 3 | 5 |

This is then transformed into vectors in Rn depending on the cipher parameters. Assume a Hill-3 Cipher is being used. The word is transformed into 3-tuple vectors. Any missing characters are pumped with a space.

Let the matrix A be whose determinant is -10, which has no common prime factors with 27 and is thus a valid matrix.

To encipher the message, the vectors created from the message are collated into a single matrix X. The ciphertext comes from the product Y = AX mod 27, where the column vectors of Y are the corresponding vectors of the ciphertext.

Which translates to

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 21 | 17 | 14 | 4 | 1 | 20 | 21 | 20 | 13 |
| U | Q | N | D | A | T | U | T | M |

So the corresponding ciphertext for BEATRICE in this cipher scheme is UQNDATUTM.

**Deciphering**

To decipher the ciphertext, first we must get the inverse of the enciphering matrix A. This is done slightly differently than the regular method. First, the adjunct of the matrix is computed. This is simply the transpose of the matrix containing all the matrix’s cofactors.

For the matrix above, the adjunct is

Normally, the inverse is found by multiplying the adjunct by the multiplicative inverse of the determinant, but inverse is now computed by multiplying the adjunct with the multiplicative modular inverse of the determinant. The multiplicative modular inverse of a number modulo m is a number a-1 such that aa-1 = 1 mod m.

The determinant of the original matrix was -10. The multiplicative modular inverse is then 8 since -10(8) = 80 = 1 mod 27.

The inverse is then

So taking the ciphertext UQNDATUTM, translating into

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| U | Q | N | D | A | T | U | T | M |
| 21 | 17 | 14 | 4 | 1 | 20 | 21 | 20 | 13 |

Which is converted into the matrix.

The plaintext is gotten by multiplying A-1Y mod 27.

Which translates to

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 20 | 18 | 9 | 3 | 5 |
| B | E | A | T | R | I | C | E |

Which is the original string from the previous chapter.

**Breaking the Cipher**

Given a plaintext “BEATRICE “ and a ciphertext “UQNDATUTM”, the original enciphering matrix could be derived.

Let p1, p2, …, pn be linearly independent plaintext vectors and c1, c2, …, cn be the corresponding ciphertext vectors, the deciphering matrix could be derived by forming first matrices C and P, C having c1T, c2T, …, cnT as row vectors and P having p1T, p2T, …, pnT as row vectors, augmenting P to C, forming [C|P] and row reducing that matrix to [In|(A-1)T]. A-1 is the transpose of the right half of the augmented matrix.

So using the same example, are linearly independent plaintext vectors and are the corresponding ciphertext vectors.

[C|P] is then formed by

Row reducing…

Getting the transpose of the right part of the augmented matrix, we have, which is the exact same deciphering matrix that was computed in the deciphering section of this chapter. The inverse it then the same as the enciphering matrix used in the beginning of the chapter, which is , proving that this algorithm remains consistent.

**IV. Design**

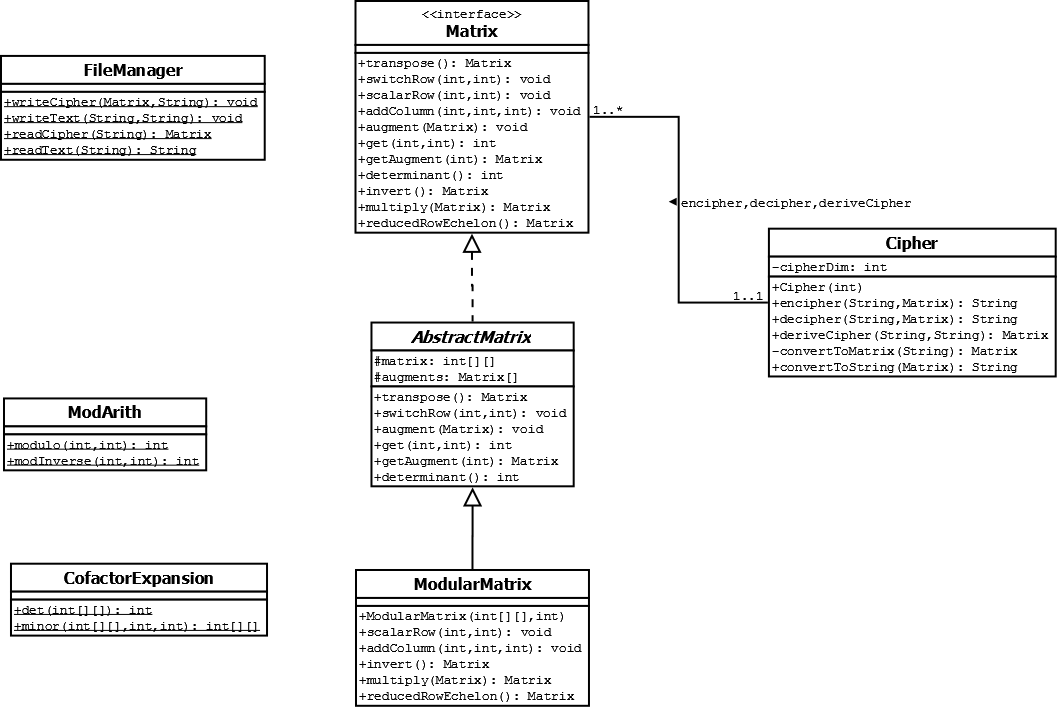
**String Representation**

The way strings are represented uses the American Standard Code for Information Interchange ASCII values of the characters. Not wanting to map to the control characters (ASCII 00h to 1Fh and 7Fh), this gives 95 mappable characters. Including the horizontal tab (09h) and the line feed (0Ah) characters; this gives 97 characters, which makes the modulus for the algorithm discussed above 97, a prime number. Thus, any matrix with a nonzero determinant which is not divisible by 97 would be usable in the Hill cipher algorithm.

The character mapping uses the following function:

**Software Architecture**

The software follows the model view architecture. The only point of interest is the model component.



The model module consists of five parts, namely Matrix, CofactorExpansion, ModArith, Cipher, and FileManager.

The Matrix part handles all matrix operations including transposition, row operations, computing for determinants, creating augmented matrices, multiplication, and reduction to reduced row echelon form. For augmented matrices, the Observer pattern was used. Every time a row operation was performed, the same operations would be performed on the augmented matrices.

Furthermore, the CofactorExpansion class simply handles computation of determinants using the cofactor expansion algorithm, which has a big-O time complexity of O(n!) given an n x n matrix.

In addition to those classes, the ModArith class handles modular arithmetic functions including modulo and finding the modular inverse.

Moreover, the Cipher class handles all Hill Cipher related algorithms, maintaining the dimension as an internal state. The operations include enciphering, deciphering, and cracking the cipher.

Finally, the FileManager class handles all input and output of ciphers, plaintext, and ciphertext.

**V. Implementation**

The Java programming language was used, where Java Swing was used for the Graphical User interface.

No special functions were used aside from the common expression operators, particularly %, which was used to compute the mathematical modulo of numbers for use in the computations using the following formula:

To find modular multiplicative inverses, the following pseudocode was followed

getModInverse(a,m) returns a modular inverse of a

Initialize aInv to 1

while a \* aInv is not congruent to 1 mod m

aInv++

return aInv

For the ModularMatrix, the superclass’ constructor is called and each value is then taken the modulo of relative to 97. All operations on this type of matrix end with this operation to maintain modulus 97.

**VI. Conclusion**

The objective of the project is to implement the enciphering, deciphering, and the breaking of the Hill Cipher. After multiple rounds of testing the final project, it can therefore be concluded that the group was successfully able to fulfill the objective. The software successfully enciphers, deciphers, and cracks the Hill-2 and Hill-3 cipher.

**VII. References**

Anton, H. & Rorres, C. (2010). *Elementary linear algebra: Applications version*. John Wiley & Sons, NJ:

New Jersey.

Cryptography. (n.d.). Retrieved November 5, 2015, In Wikipedia: https://en.wikipedia.org/wiki

/Cryptography

**Appendix A – Test Cases**

|  |  |  |
| --- | --- | --- |
| **User Story ID / Title:** | | 1 / The user can encipher a string to increase security |
| **Type of Test** | | Black box |
| **Purpose of Test** | | Check if the user can input a string and encipher it accordingly |
| **Test Case Number** | | 1-1 |
|  | **Objective of Case** | Check if the string can be enciphered using Hill 2 |
|  | **Input** | Cipher Type: Hill 2  \*user presses Set Cipher, then inputs the following\*  1, 4, 5, 3  \*user presses Set Cipher on the new window\*  Plaintext: EXCALIBUR |
|  | **Expected Result** | Ciphered text: c^fp.TTFRX |
|  | **Actual Result** | Ciphered text: c^fp.TTFRX |
|  | **Conclusion** | The program was able to encipher the string correctly. |
|  | **Remarks** |  |
|  | **Tester** | Syfu, Jonah |
| **Test Case Number** | | 1-2 |
|  | **Objective of Case** | Check if the string can be enciphered using Hill 3 |
|  | **Input** | Cipher Type: Hill 3  \*user presses Set Cipher, then inputs the following\*  2, 3, 6, 7, 3, 4, 2, 12, 1  \*user presses Set Cipher on the new window\*  Plaintext: EXCALIBUR |
|  | **Expected Result** | Ciphered text: `r%XJUJ/k |
|  | **Actual Result** | Ciphered text: `r%XJUJ/k |
|  | **Conclusion** | The program was able to cipher the string correctly. |
|  | **Remarks** |  |
|  | **Tester** | Syfu, Jonah |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 1-3 |
|  | **Objective of Case** | Check if the cipher can be changed |
|  | **Input** | Cipher Type: Hill 3  \*user presses Set Cipher, then inputs the following\*  1, 3, 5, 7, 4, 2, 4, 1, 9  \*user presses Set Cipher on the new window\*  \*user presses Set Cipher once more, then inputs the following\*  1, 3, 5, 7, 4, 2, 4, 12, 2  \*user presses Set Cipher on the new window\* |
|  | **Expected Result** | New Cipher: 1, 3, 5, 7, 4, 2, 4, 12, 2 |
|  | **Actual Result** | New Cipher: 1, 3, 5, 7, 4, 2, 4, 12, 2 |
|  | **Conclusion** | The program was able to change the cipher. |
|  | **Remarks** |  |
|  | **Tester** | Syfu, Jonah |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 1-4 |
|  | **Objective of Case** | Check if the Clear Cipher button is working |
|  | **Input** | Cipher Type: Hill 2  \*user presses Set Cipher, then inputs the following\*  1, 4, 5, 3  \*user presses Set Cipher on the new window\*  \*user presses Clear Cipher\* |
|  | **Expected Result** | Cipher: \*empty\* |
|  | **Actual Result** | Cipher: 1, 4, 5, 3 |
|  | **Conclusion** | The cipher was not cleared. However, upon further checking, the cipher was indeed cleared, just not reflected in the GUI. |
|  | **Remarks** | Perhaps have the GUI update accordingly when the cipher has been cleared. |
|  | **Tester** | Syfu, Jonah |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 1-5 |
|  | **Objective of Case** | Check if the Clear Text button is working |
|  | **Input** | Plaintext: EXCALIBUR  \*user presses Clear Text\* |
|  | **Expected Result** | Plaintext: \*empty\* |
|  | **Actual Result** | Plaintext: \*empty\* |
|  | **Conclusion** | The program was able to clear the text. |
|  | **Remarks** |  |
|  | **Tester** | Syfu, Jonah |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 1-6 |
|  | **Objective of Case** | Check if the Clear All button is working |
|  | **Input** | Cipher Type: Hill 2  \*user presses Set Cipher, then inputs the following\*  1, 4, 5, 3  \*user presses Set Cipher on the new window\*  Plaintext: EXCALIBUR  \*user presses Clear Text\* |
|  | **Expected Result** | Cipher: \*empty\*  Plaintext: \*empty\* |
|  | **Actual Result** | Cipher: 1, 4, 5, 3  Plaintext: \*empty\* |
|  | **Conclusion** | The program was not able to clear the cipher. But upon further checking, the cipher was indeed cleared, just not reflected in the GUI. |
|  | **Remarks** |  |
|  | **Tester** | Syfu, Jonah |

|  |  |  |
| --- | --- | --- |
| **User Story ID / Title:** | | 2 / User can choose to decipher ciphertext to discover the original plaintext |
| **Type of Test** | | Black box |
| **Purpose of Test** | | Check if user can decipher a ciphertext properly |
| **Test Case Number** | | 2-1 |
|  | **Objective of Case** | Check if the string can be deciphered using Hill 2 |
|  | **Input** | Cipher Type: Hill - 2  \*user presses Set Cipher, then inputs the following\*  1,4,5,3  \*user presses Set Cipher on the new window\*  Cipher Text: c^fp.TTFRX |
|  | **Expected Result** | EXCALIBUR |
|  | **Actual Result** | EXCALIBUR |
|  | **Conclusion** | The program was able to decipher the text properly. |
|  | **Remarks** | Word wrap the CipherText. The program would decipher the cipher text even if a hill - 3 matrix is the one loaded or set. |
|  | **Tester** | Andres, John Joseph |
| **Test Case Number** | | 2-2 |
|  | **Objective of Case** | Check if the string can be deciphered using Hill 3 |
|  | **Input** | Cipher Type: Hill - 3  \*user presses Set Cipher, then inputs the following\*  2, 3, 6, 7, 3, 4, 2, 12, 1  \*user presses Set Cipher on the new window\*  Cipher Text: `r%XJUJ/k |
|  | **Expected Result** | EXCALIBUR |
|  | **Actual Result** | EXCALIBUR |
|  | **Conclusion** | The program was able to decipher the text properly |
|  | **Remarks** | Word wrap the CipherText. The program would decipher the cipher text even if a hill - 2 matrix is the one loaded. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-3 |
|  | **Objective of Case** | Check if the cipher can be changed |
|  | **Input** | Cipher Type: Hill - 3  \*user presses Set Cipher, then inputs the following\*  2, 3, 6, 7, 3, 4, 2, 12, 1  \*user presses Set Cipher on the new window\*  \*user presses Set Cipher once more, then inputs the following\*  2, 4, 5, 6, 1, 3, 4, 7, 69  \*user presses Set Cipher on the new window\* |
|  | **Expected Result** | New Cipher: 2, 4, 5, 6, 1, 3, 4, 7, 69 will be displayed |
|  | **Actual Result** | New Cipher: 2, 4, 5, 6, 1, 3, 4, 7, 69 is displayed |
|  | **Conclusion** | The program was able to change the cipher. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-4 |
|  | **Objective of Case** | Check if the Clear Cipher button will work |
|  | **Input** | Cipher Type: Hill - 2  \*user presses Set Cipher, then inputs the following\*  69, 2, 44, 3  \*user presses Set Cipher on the new window\*  \*user presses Clear Cipher\* |
|  | **Expected Result** | Cipher: 69, 2, 44, 3 will be removed from the main window |
|  | **Actual Result** | Cipher: 69, 2, 44, 3 did not disappear from the main window |
|  | **Conclusion** | The software did not delete the Cipher from the Interface but upon clicking the Decipher button it is shown that the Cipher was cleared in the system. |
|  | **Remarks** | Have the function Delete Cipher reflect on the Interface. Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-5 |
|  | **Objective of Case** | Check if the Clear Text button is working |
|  | **Input** | Cipher Text: High School Musical is Best Movie Ever  \*user presses Clear Text\* |
|  | **Expected Result** | Enter ciphertext here… is displayed |
|  | **Actual Result** | Enter ciphertext here… is displayed |
|  | **Conclusion** | The program was able to clear the text. |
|  | **Remarks** | Works the same for both Cipher Type. Word wrap the ciphertext. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-6 |
|  | **Objective of Case** | Check if the Clear All button is working |
|  | **Input** | Cipher Type: Hill 3  \*user presses Set Cipher, then inputs the following\*  1,2,3,4,5,6  \*user presses Set Cipher on the new window\*  CipherText: Hum Hallelujah  \*user presses Clear All\* |
|  | **Expected Result** | Enter ciphertext here … is displayed  Cipher: 69, 2, 44, 3 will be removed from the main window |
|  | **Actual Result** | Enter ciphertext here … is displayed  Cipher: 69, 2, 44, 3 is not removed from the main window /Fixed/ |
|  | **Conclusion** | The software did not delete the Cipher from the Interface but upon clicking the Decipher button it is shown that the Cipher was cleared in the system. The ciphertext was successfully removed. |
|  | **Remarks** | Have the function Delete Cipher reflect on the Interface./fixed/ Works the same for both Cipher Type. Word wrap the CipherText. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-7 |
|  | **Objective of Case** | Check if the Decipher Button will work if there are no loaded Ciphers |
|  | **Input** | Cipher Type: Hill 2  CipherText: Hum Hallelujah  \*user presses Clear All\* |
|  | **Expected Result** | Error Message will display |
|  | **Actual Result** | Error Message “Please input ciphertext” was displayed. |
|  | **Conclusion** | The software will not allow user to decipher without a cipher |
|  | **Remarks** | Works the same for both Cipher Type. Word wrap the CipherText. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-8 |
|  | **Objective of Case** | Check if the Set Cipher coordinates will accept other inputs that are not non-negative integers. |
|  | **Input** | Cipher Type: Hill 2  CipherText: Hum Hallelujah  \*user presses Clear All\* |
|  | **Expected Result** | Error Message “Input Format Error” will display. |
|  | **Actual Result** | Error Message “Input Format Error” was displayed. |
|  | **Conclusion** | The software will not allow user to input letters and special characters but a negative number will become a different positive number. |
|  | **Remarks** | Works the same for both Cipher Type. Word wrap the CipherText. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-9 |
|  | **Objective of Case** | Check if system saves cipher file . |
|  | **Input** | Cipher Type: Hill 2  \*User selects set Cipher and inputs cipher\*  Cipher: 1, 4, 5, 6 \*User selects set Cipher and Cipher appears on screen\* \*User selects save Cipher\* Save Cipher file as Cipher2Try \*User selects save\* |
|  | **Expected Result** | Cipher file Cipher2Try.hill is saved into designated location. |
|  | **Actual Result** | Cipher file Cipher2Try.hill is saved into designated location. |
|  | **Conclusion** | The software has successfully saved the Cipher file. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-10 |
|  | **Objective of Case** | Check if system ensures loaded cipher file is a cipher file and is loaded correctly. |
|  | **Input** | Cipher Type: Hill 2  \*User selects Load Cipher and chooses a cipher they wish to load\* \*User selects Cipher2Try.hill and clicks open\* |
|  | **Expected Result** | Cipher2Try.hill is successfully loaded and the Cipher: 1, 4, 5, 6 is displayed on screen. |
|  | **Actual Result** | Cipher2Try.hill is successfully loaded and the Cipher: 1, 4, 5, 6 is displayed on screen. |
|  | **Conclusion** | The software has successfully loaded the Cipher file. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-11 |
|  | **Objective of Case** | Check if system can successfully load cipher text File |
|  | **Input** | Cipher Type: Hill 2  \*User selects Load CipherText File and chooses a text file they wish to decipher\* \*User selects CipherTextTry.txt and clicks open\* |
|  | **Expected Result** | CipherTextTry.txt is successfully loaded and the text cDfr.nT#RX is displayed on screen. |
|  | **Actual Result** | CipherTextTry.txt is successfully loaded and the text cDfr.nT#RX is displayed on screen. |
|  | **Conclusion** | The software has successfully loaded the Cipher text file. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 2-12 |
|  | **Objective of Case** | Check if system can successfully save PlainText File |
|  | **Input** | Cipher Type: Hill 2  \*User selects Save PlainText File and names the File Excalibur.txt then saves\* |
|  | **Expected Result** | Excalibur.txt is successfully saved in the desired location. |
|  | **Actual Result** | Excalibur.txt is successfully saved in the desired location. |
|  | **Conclusion** | The software has successfully saved the plaintext file. |
|  | **Remarks** | Works the same for both Cipher Type.. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **User Story ID / Title:** | | 3 / User can choose to derive the enciphering matrix given plaintext and ciphertext |
| **Type of Test** | | Black box |
| **Purpose of Test** | | Check if user can derive a cipher properly. |
| **Test Case Number** | | 3-1 |
|  | **Objective of Case** | Check if system can derive a cipher from loaded text file and cipher file. |
|  | **Input** | Cipher Type: Hill-2  \*User loads text file EXCALIBUR.txt\* \*User loads cipher file CipherTextTry.txt\*  \*User clicks Derive Cipher\* |
|  | **Expected Result** | Cipher: 1, 4, 5, 6 |
|  | **Actual Result** | Cipher: 1, 4, 5, 6 is displayed on screen |
|  | **Conclusion** | The system can properly derive a cipher using loaded text files. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-2 |
|  | **Objective of Case** | Check if system can load ciphertext and text file correctly |
|  | **Input** | Cipher Type: Hill-2  \*User loads Cipher text file Excalibur.txt\* \*User loads text file CipherTextTry.txt\* |
|  | **Expected Result** | the text EXCALIBUR and cDfr.nT#RX are displayed on screen. |
|  | **Actual Result** | the text EXCALIBUR and cDfr.nT#RX are displayed on screen. |
|  | **Conclusion** | The system can properly load a plaintext file and a ciphertext file. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-3 |
|  | **Objective of Case** | Check if system can derive cipher properly. |
|  | **Input** | Cipher Type: Hill-2  \*User inputs Ducks on plaintext\* \*User inputs ,\70sW on ciphertext\* |
|  | **Expected Result** | Cipher: 1, 2, 3, 4 is derived |
|  | **Actual Result** | Cipher: 1, 2, 3, 4 is displayed on screen |
|  | **Conclusion** | The system successfully displayed the expected cipher on screen. |
|  | **Remarks** | Works the same for both Cipher Type. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-4 |
|  | **Objective of Case** | Check if the Clear Text button is working |
|  | **Input** | Cipher Text: High School Musical is Best Movie Ever  \*user presses Clear Text\* |
|  | **Expected Result** | Enter ciphertext here… is displayed |
|  | **Actual Result** | Enter ciphertext here… is displayed |
|  | **Conclusion** | The program was able to clear the text. |
|  | **Remarks** | Works the same for both Cipher Type. Word wrap the ciphertext. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-5 |
|  | **Objective of Case** | Check if the Clear All button is working |
|  | **Input** | Cipher Type: Hill 3  \*user presses Set Cipher, then inputs the following\*  1,2,3,4,5,6  \*user presses Set Cipher on the new window\*  CipherText: Hum Hallelujah  \*user presses Clear All\* |
|  | **Expected Result** | Enter ciphertext here … is displayed.  Cipher: 69, 2, 44, 3 will be removed from the main window. |
|  | **Actual Result** | Enter ciphertext here … is displayed.  Cipher: 69, 2, 44, 3 is removed from the main window. |
|  | **Conclusion** | The cipher was successfully removed. The ciphertext was successfully removed. |
|  | **Remarks** | Works the same for both Cipher Type. Word wrap the CipherText. |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-6 |
|  | **Objective of Case** | Check if the Home button is working |
|  | **Input** | \*user presses Home\* |
|  | **Expected Result** | Screen would display the main menu |
|  | **Actual Result** | Screen would display the main menu |
|  | **Conclusion** | The home button works correctly. |
|  | **Remarks** | None |
|  | **Tester** | Andres, John Joseph |

|  |  |  |
| --- | --- | --- |
| **Test Case Number** | | 3-7 |
|  | **Objective of Case** | Check if the derive cipher button will display without the plain text or cipher text |
|  | **Input** | \*User presses the Derive Cipher button\* |
|  | **Expected Result** | An error message should display. |
|  | **Actual Result** | The error message no Cipher Text has been displayed. |
|  | **Conclusion** | The Derive function will only work when there is an input from the user. |
|  | **Remarks** | None |
|  | **Tester** | Andres, John Joseph |