|  |  |  |
| --- | --- | --- |
| **ipn** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO** |  |

**Cryptography**

**“Affine cipher”**

Abstact

The following report is an implementation of the Affine cipher to encrypt and decrypt a message using Java as the programable language.

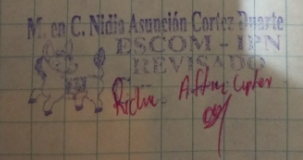
**By:**

**Ricardo Vargas Sagrero**

Professor:

MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

September 2018



To validate this report it is necessary to include the corresponding seal

**Index**

Contenido

[Introducción 1](#_Toc525426585)

[Revisión de literatura 1](#_Toc525426586)

[Software 3](#_Toc525426587)

[Procedimiento 3](#_Toc525426588)

[Resultados 5](#_Toc525426589)

[Discusión 8](#_Toc525426590)

[Conclusión 8](#_Toc525426591)

[Referencias 8](#_Toc525426592)

[Código 9](#_Toc525426593)

# Introducción

La criptografía tiene sus inicios desde que se vio la necesidad de esconder ciertos mensajes de todas las personas, con el fin de que solo el emisor y el receptor pudieran entender su significado. El origen de la criptografía podría decirse que viene desde que se creo la escritura, ya que con ella se comenzó a dejar escrito todo lo que pasaba en las civilizaciones y así cualquier que pudiera leer o comprender la escritura podría saber que era lo que significa. Con el tiempo y la generalización de la lectura se vieron los riesgos que implicaba que la información fuera leída por personas ajenas, por ello se crearon los primeros métodos de protección de lo escrito. Habitualmente la información a proteger se le conoce como mensaje en claro y la información oculta se le conoce como mensaje cifrado. En este trabajo se hablará acerca del método de Afiine cipher, un método mono alfabético para cifrar un mensaje en claro por medio de un corrimiento en el abecedario. A continuación, se deberá la literatura del cifrador.

# Revisión de literatura

El cifrador affine cipher es un método para cifrar un mensaje mono alfabético, esto quiere decir que para cada letra habrá una respectiva en otro abecedario. El método recibe como parámetro α (alfa), β(beta) y N(Tamaño del abecedario) con el fin de hacer esto corrimiento mono alfabético. Todas las entradas deben numéricos. Para este trabajo se contara con un ya que se utilizara el abecedario en ingles, esto nos a describir las restricciones para el método.

1. El Máximo común divisor de α y N debe ser igual a 1.
2. El número α deberá estar contenido entre 0 y n

Para cifrar un mensaje se utiliza la siguiente función

Donde:

Para descifrar el mensaje se despejará la función de cifrado para encontrar la función que descifre el mensaje, suponiendo que el valor de α = 3 y β = 5:

Aplicando el módulo 26 en la ecuación podemos encontrar el inverso aditivo de numero 5, esto se hace con la siguiente ecuación:

Por lo tanto, se debe encontrar un número que al sumarlo con β = 5 su modulo debe ser igual a 0.

Sustituimos el valor de la ecuación 2 en la ecuación 1, lo que nos da el siguiente resultado

Lo que sigue es encontrar el inverso multiplicativo de 3, por lo cual utilizaremos la siguiente ecuación:

El número por su inverso multiplicativo modulo n debe ser igual a 1. Por lo que buscamos que un número que multiplicado por α modulo 26 sea 1:

El número encontrado fue 9 por lo que la función de descifrar queda de la siguiente forma

Para reducir la expresión podemos aplicar el módulo 26 a 180 por lo que la función final de descifrado queda de la siguiente formado:

Lo que hace mucho más sencillo al momento de hacer los cálculos con todas las entradas.

# Software

Para este trabajo se utilizaron las siguientes herramientas

* Entorno de desarrollo integrado: Netbeans
* Lenguaje de programación: Java
* Sistema operativo: Windows 10
* Bibliotecas estándar de entrada y salida de Java
* JFrame para la interfaz grafica

# Procedimiento

Procedimiento para cifrar un mensaje:

Mensaje Cifraddo

Función de cifrado

Mensaje en claro

Ejemplo de cifrador affine ciffer, cifrar el mensaje cryptography usando el abecedario en inglés, alpha = 3 y beta 5:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| K | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| U | **V** | **W** | **X** | **Y** | **Z** |  |  |  |  |
| 20 | 21 | 22 | 23 | 24 | 25 |  |  |  |  |

Tabla . Abecedario Ingles

|  |  |  |  |
| --- | --- | --- | --- |
| Mensaje en claro | Formula | Resultado | Mensaje cifrado |
| C |  | 11 | L |
| R |  | 4 | E |
| Y |  | 25 | Z |
| P |  | 24 | Y |
| T |  | 10 | K |
| O |  | 21 | V |
| G |  | 23 | X |
| R |  | 4 | E |
| A |  | 5 | F |
| P |  | 24 | Y |
| H |  | 0 | A |
| Y |  | 25 | Z |

Tabla . Mensaje cifrado

El mensaje cifrado es ‘LEZYKVXEFYAZ’ por lo que la comprensión del texto el claro ya no es directa.

Ahora se procederá a descifrar el mensaje utilizando la formula obtenida en el apartado de revisión de literatura para descifrar el mensaje ya que se cuenta con los mismo Alpha y beta:

|  |  |  |  |
| --- | --- | --- | --- |
| Mensaje cifrado | Formula | Resultado | Mensaje en claro |
| L |  | 2 | c |
| E |  | 17 | r |
| Z |  | 24 | y |
| Y |  | 15 | p |
| K |  | 19 | t |
| V |  | 14 | o |
| X |  | 6 | g |
| E |  | 17 | r |
| F |  | 0 | a |
| Y |  | 15 | p |
| A |  | 7 | h |
| Z |  | 24 | y |

El mensaje en claro es “Cryptography”.

# Resultados

Se utilizo un JFrame para este trabajo, quedando como resultado lo siguiente:

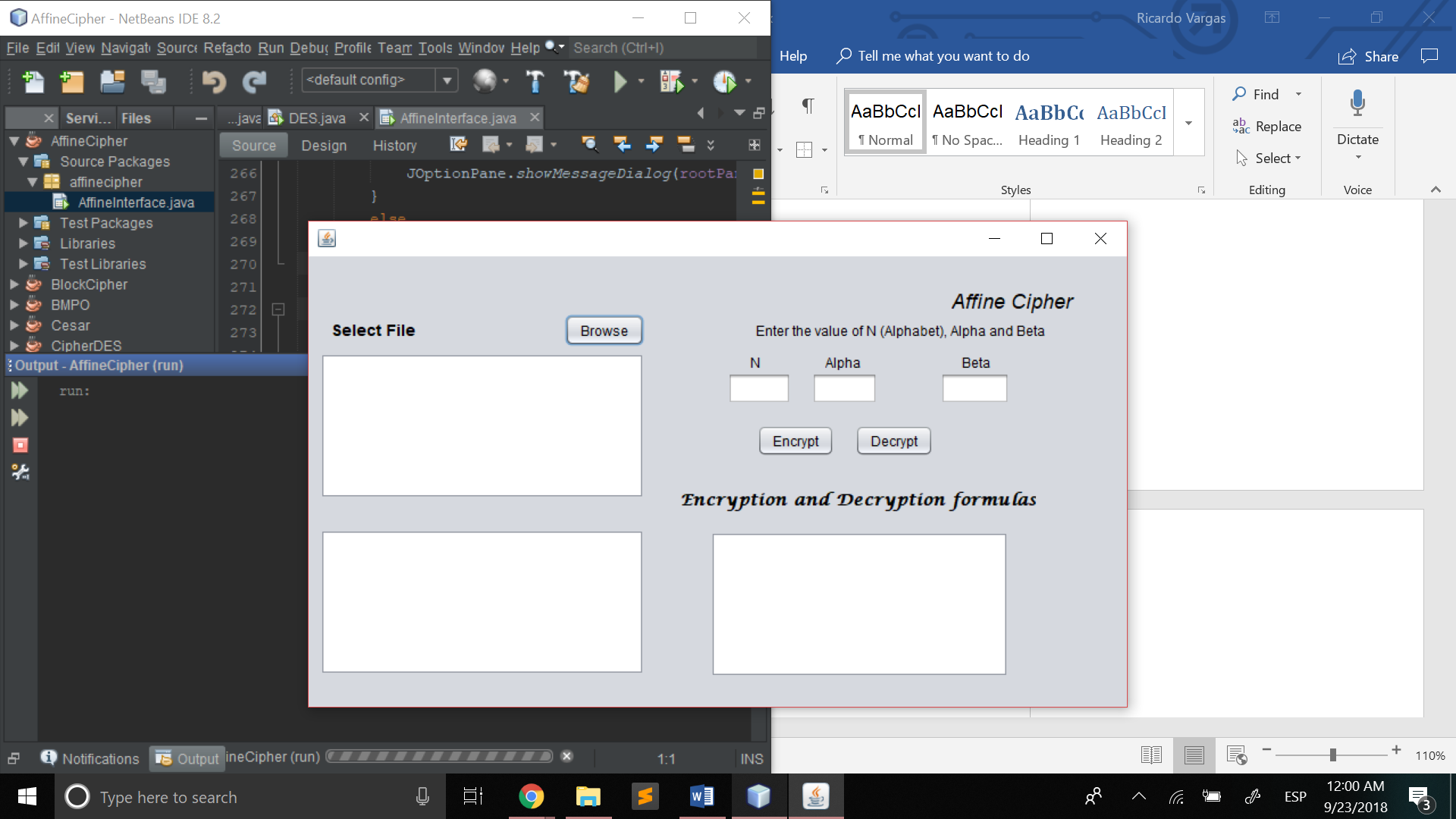


Ilustración . Inicialización del programa

Dando clic en el botón “Browse” se abrirá una pantalla la cual nos permitirá seleccionar un archivo como se ilustra en la siguiente imagen:

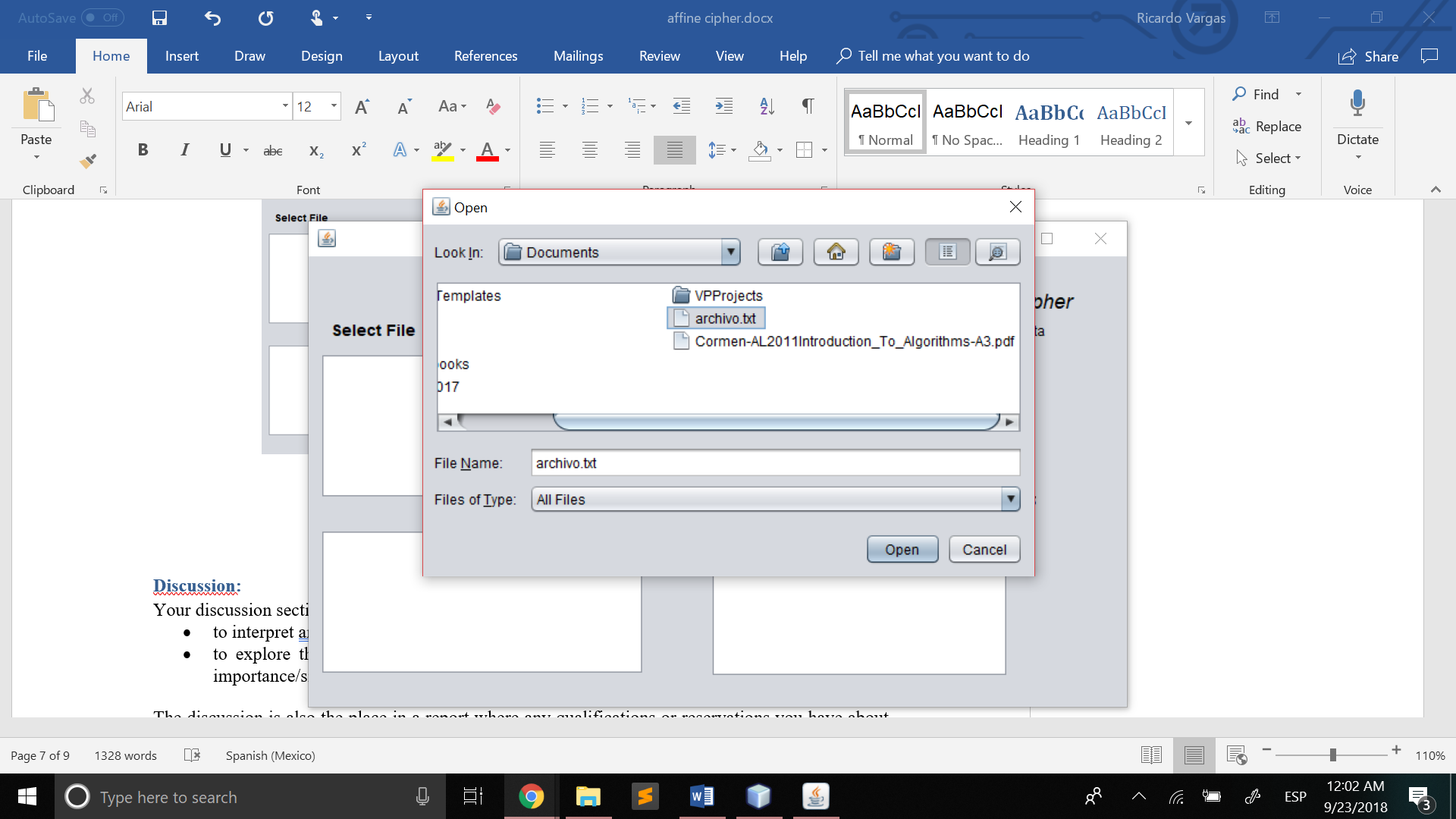


Ilustración . Seleccionar archivo

Una vez seleccionado el archivo, el programa cargara la información del texto del archivo y la imprimirá en el primer recuadro, quedando de la siguiente forma:

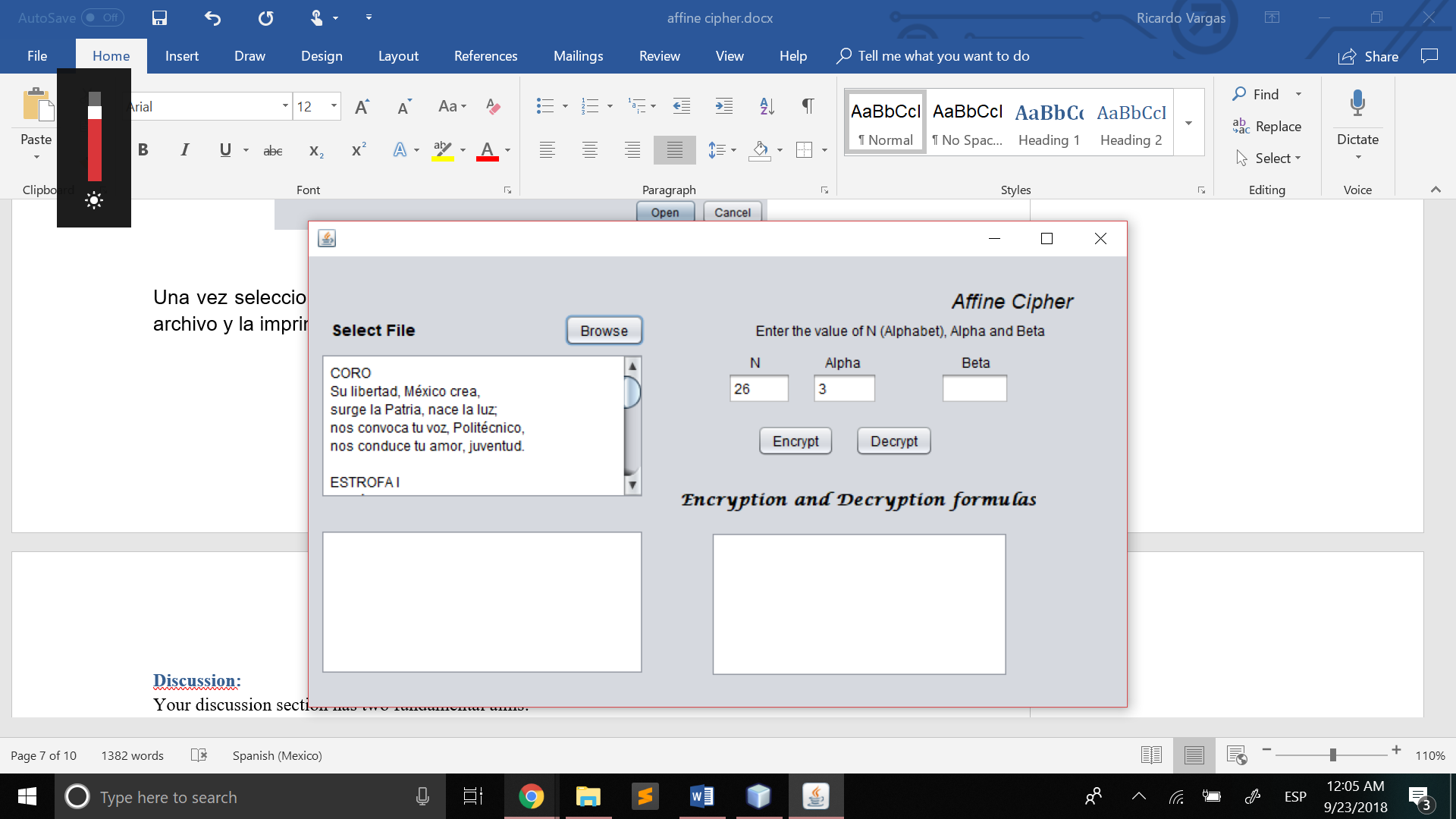


Ilustración . Texto seleccionado

Lo que sigue es ingresar el los valor de β, α y N y seleccionar el botón “Encypt”, el resultado se mostrara en la pantalla inferior izquierda y en la pantalla inferir derecha se mostrara la clave con la que el mensaje fue cifrado:

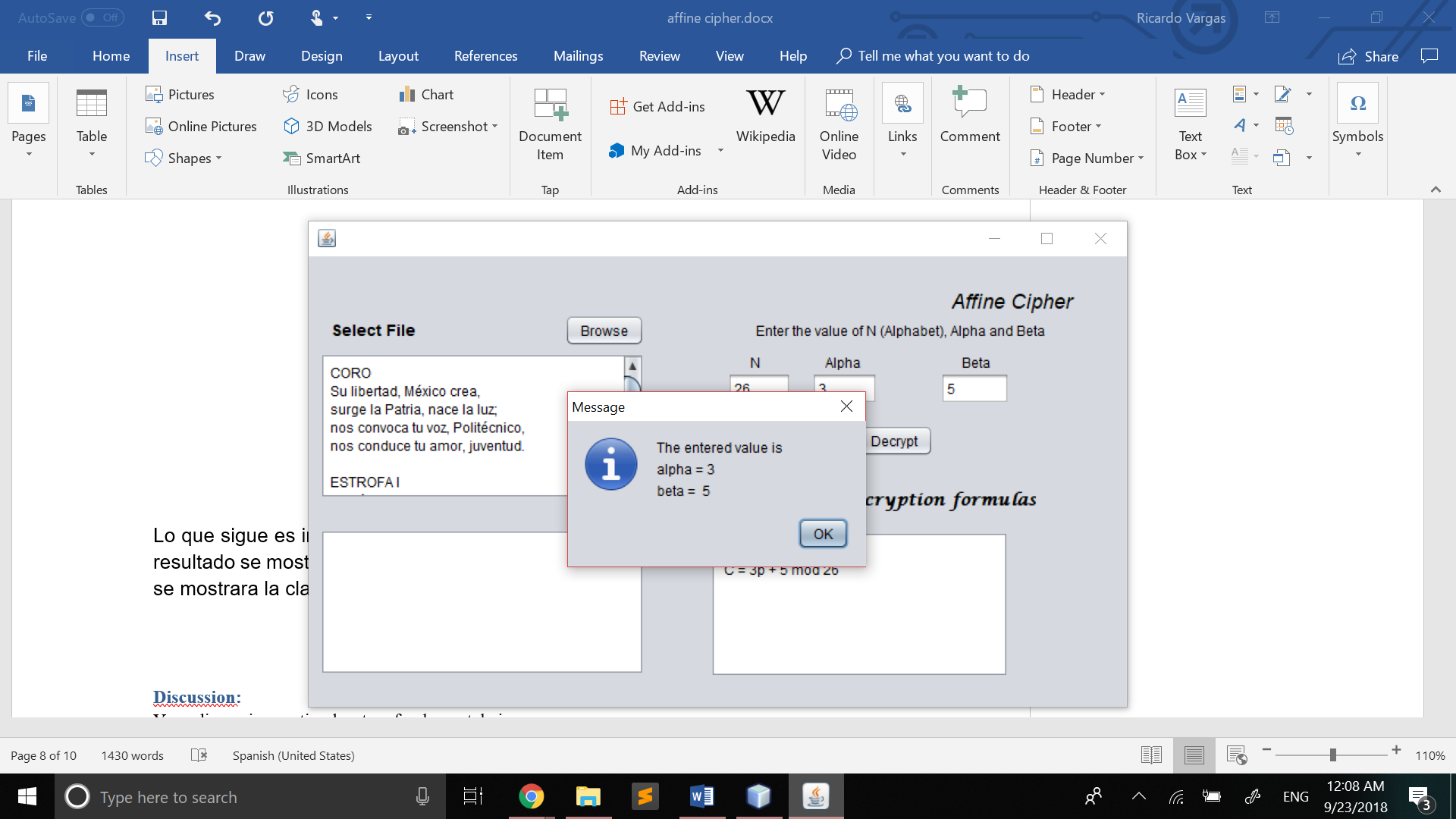


Ilustración . Valores ingresados

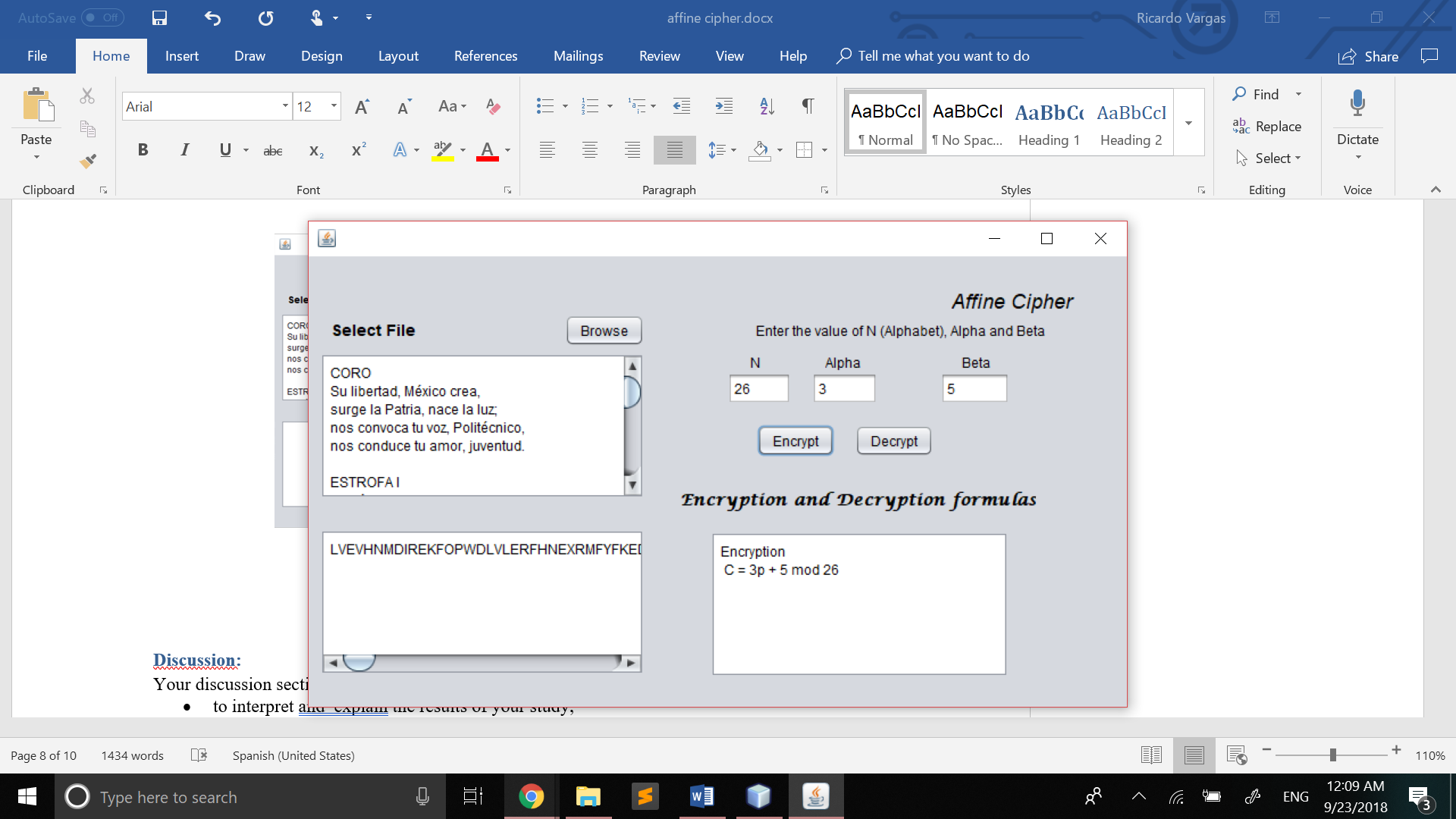


Ilustración . Mensaje cifrado

Para descifrar el mensaje lo único que se tiene que hacer es hacer clic en el botón de “Decypt” y se mostrara el mensaje descifrado en la pantalla inferior izquierda y la clave con la que fue descifrado en la esquina inferior derecha, quedando de a siguiente forma:

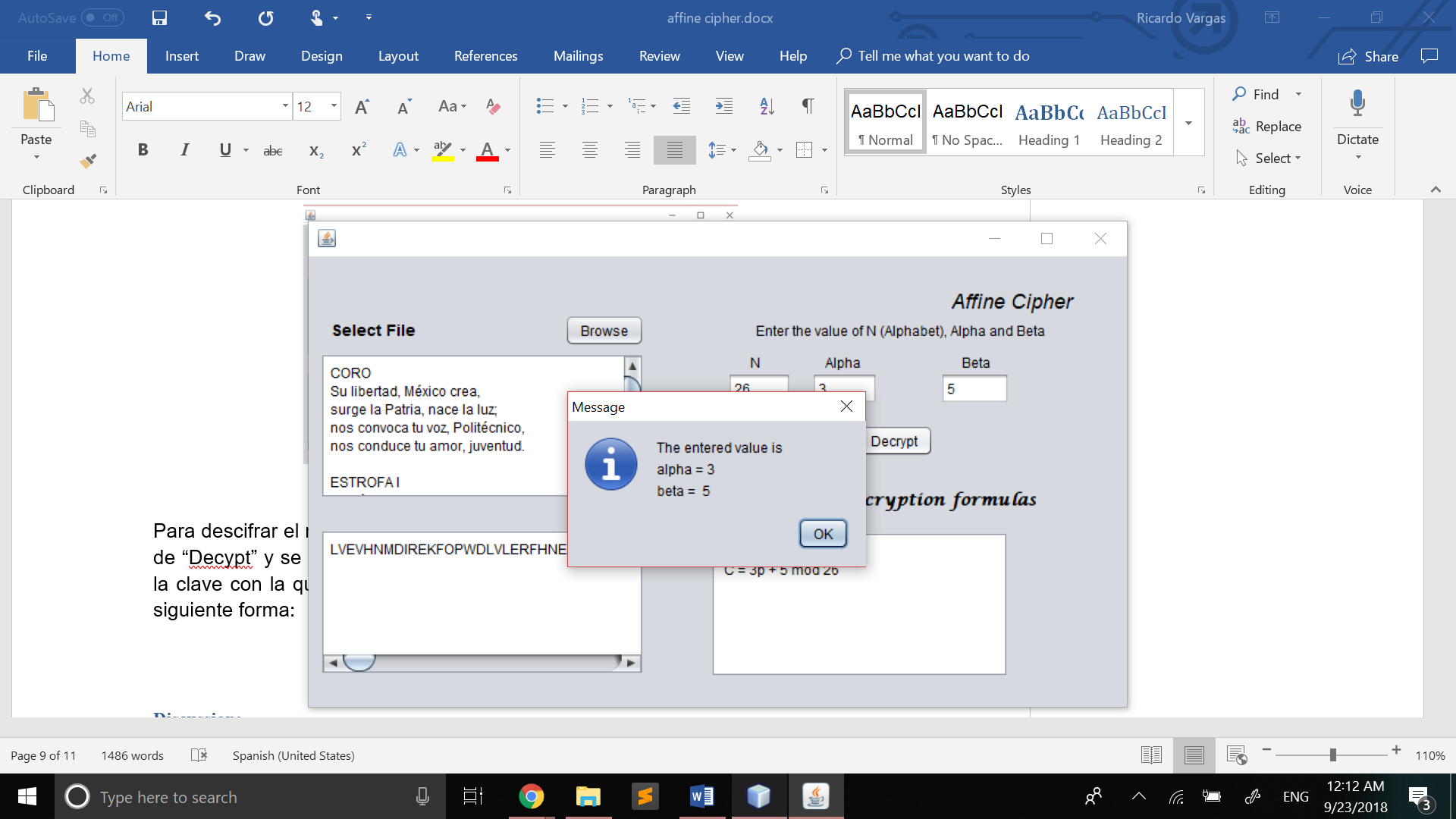


Ilustración . Valor ingresados

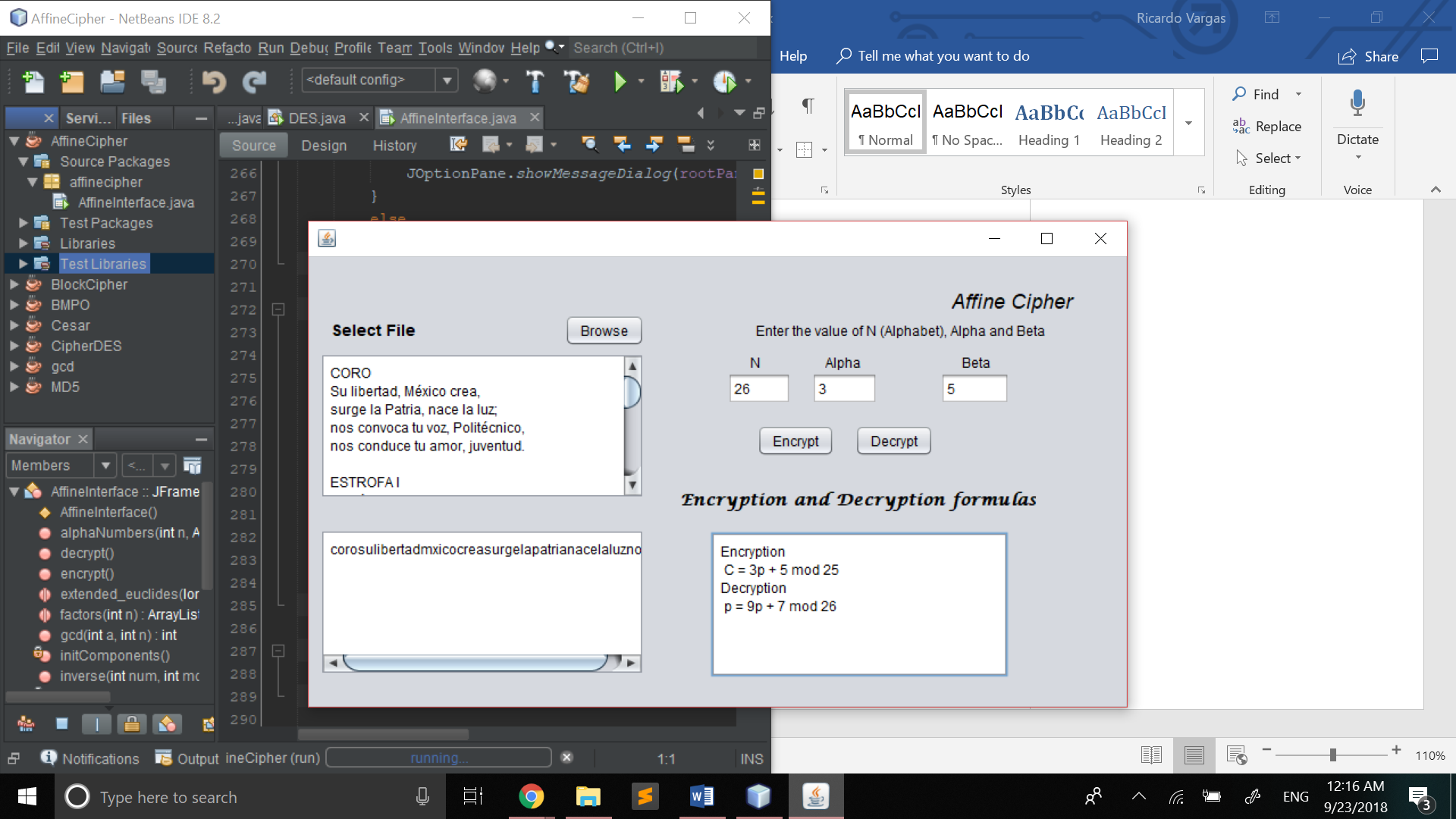


Ilustración . Resultado final

# Discusión

Los resultados de esta práctica fueron exitosos, se logró implementar la función para cifrar un mensaje usando affine cipher. Con base a la teoría matemática de esta función se vieron algunas variantes de este programa y sus límites. Por lo que en la siguiente practica se mejorara el funcionamiento de esta.

# Conclusión

Se tuvo problemas con este trabajo al momento de ingresar valores que no pertenecían al anillo de Z ya que solo en los valores de Alpha en los cuales se cumple la ecuación se podía cifrar el mensaje correctamente, cuando se ingresa un número que no pertenecía se tenía que mostrar un error. Para este trabajo se tuvo que ingresar manualmente los valores del anillo Z para que se hiciera la comprobación del máximo común divisor y así poder cifrar con éxito el mensaje.

# Referencias

1. Theoretically (2015). Affine Cipher - Decryption (Known Plaintext Attack). [video] Available at: https://www.youtube.com/watch?v=ry3g0xN8QKU [Accessed 3 Sep. 2018].
2. Rodriguez, D. (2013). Affine Cipher. [online] Crypto Corner. Available at: http://crypto.interactive-maths.com/affine-cipher.html [Accessed 3 Sep. 2018].

It is essential to include a reference list or bibliography of the reference material you

# Código

|  |
| --- |
| 1. /\*\* 2. \* 3. \* @author rsagr 4. \*/ 5. **public** **class** AffineInterface **extends** javax.swing.JFrame { 7. /\*\* 8. \* Creates new form AffineInterface 9. \*/ 10. **public** AffineInterface() { 11. initComponents(); 12. } 14. /\*\* 15. \* This method is called from within the constructor to initialize the form. 16. \* WARNING: Do NOT modify this code. The content of this method is always 17. \* regenerated by the Form Editor. 18. \*/ 19. @SuppressWarnings("unchecked") 20. // <editor-fold defaultstate="collapsed" desc="Generated Code"> 21. **private** **void** initComponents() { 23. jLabel1 = **new** javax.swing.JLabel(); 24. jLabel2 = **new** javax.swing.JLabel(); 25. jButton1 = **new** javax.swing.JButton(); 26. jScrollPane1 = **new** javax.swing.JScrollPane(); 27. jTextArea1 = **new** javax.swing.JTextArea(); 28. jLabel3 = **new** javax.swing.JLabel(); 29. jLabel4 = **new** javax.swing.JLabel(); 30. jTextField1 = **new** javax.swing.JTextField(); 31. jTextField2 = **new** javax.swing.JTextField(); 32. jLabel5 = **new** javax.swing.JLabel(); 33. jLabel6 = **new** javax.swing.JLabel(); 34. jScrollPane2 = **new** javax.swing.JScrollPane(); 35. jTextArea2 = **new** javax.swing.JTextArea(); 36. jButton2 = **new** javax.swing.JButton(); 37. jButton3 = **new** javax.swing.JButton(); 38. jLabel7 = **new** javax.swing.JLabel(); 39. jTextField3 = **new** javax.swing.JTextField(); 40. jScrollPane3 = **new** javax.swing.JScrollPane(); 41. jTextArea3 = **new** javax.swing.JTextArea(); 42. jLabel8 = **new** javax.swing.JLabel(); 44. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE); 46. jLabel1.setFont(**new** java.awt.Font("Dialog", 2, 18)); // NOI18N 47. jLabel1.setText("Affine Cipher "); 49. jLabel2.setFont(**new** java.awt.Font("Dialog", 1, 14)); // NOI18N 50. jLabel2.setText("Select File"); 52. jButton1.setText("Browse"); 53. jButton1.addActionListener(**new** java.awt.event.ActionListener() { 54. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) { 55. jButton1ActionPerformed(evt); 56. } 57. }); 59. jTextArea1.setColumns(20); 60. jTextArea1.setRows(5); 61. jScrollPane1.setViewportView(jTextArea1); 63. jLabel4.setText("Enter the value of N (Alphabet), Alpha and Beta"); 65. jTextField1.addActionListener(**new** java.awt.event.ActionListener() { 66. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) { 67. jTextField1ActionPerformed(evt); 68. } 69. }); 71. jLabel5.setText("Alpha"); 73. jLabel6.setText("Beta"); 75. jTextArea2.setColumns(20); 76. jTextArea2.setRows(5); 77. jScrollPane2.setViewportView(jTextArea2); 79. jButton2.setText("Encrypt"); 80. jButton2.addActionListener(**new** java.awt.event.ActionListener() { 81. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) { 82. jButton2ActionPerformed(evt); 83. } 84. }); 86. jButton3.setText("Decrypt"); 87. jButton3.addActionListener(**new** java.awt.event.ActionListener() { 88. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) { 89. jButton3ActionPerformed(evt); 90. } 91. }); 93. jLabel7.setText("N"); 95. jTextField3.addActionListener(**new** java.awt.event.ActionListener() { 96. **public** **void** actionPerformed(java.awt.event.ActionEvent evt) { 97. jTextField3ActionPerformed(evt); 98. } 99. }); 101. jTextArea3.setColumns(20); 102. jTextArea3.setRows(5); 103. jScrollPane3.setViewportView(jTextArea3); 105. jLabel8.setFont(**new** java.awt.Font("Lucida Calligraphy", 1, 14)); // NOI18N 106. jLabel8.setText("Encryption and Decryption formulas"); 108. javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(getContentPane()); 109. getContentPane().setLayout(layout); 110. layout.setHorizontalGroup( 111. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 112. .addGroup(layout.createSequentialGroup() 113. .addContainerGap(613, Short.MAX\_VALUE) 114. .addComponent(jLabel3) 115. .addGap(106, 106, 106)) 116. .addGroup(layout.createSequentialGroup() 117. .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE) 118. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 119. .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING) 120. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup() 121. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, **false**) 122. .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 285, javax.swing.GroupLayout.PREFERRED\_SIZE) 123. .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 285, javax.swing.GroupLayout.PREFERRED\_SIZE) 124. .addGroup(layout.createSequentialGroup() 125. .addGap(10, 10, 10) 126. .addComponent(jLabel2) 127. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE) 128. .addComponent(jButton1))) 129. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 130. .addGroup(layout.createSequentialGroup() 131. .addGap(98, 98, 98) 132. .addComponent(jLabel4)) 133. .addGroup(layout.createSequentialGroup() 134. .addGap(99, 99, 99) 135. .addComponent(jButton2) 136. .addGap(18, 18, 18) 137. .addComponent(jButton3)) 138. .addGroup(layout.createSequentialGroup() 139. .addGap(58, 58, 58) 140. .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, 262, javax.swing.GroupLayout.PREFERRED\_SIZE)) 141. .addGroup(layout.createSequentialGroup() 142. .addGap(73, 73, 73) 143. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 144. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup() 145. .addComponent(jLabel7) 146. .addGap(45, 45, 45)) 147. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup() 148. .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, 56, javax.swing.GroupLayout.PREFERRED\_SIZE) 149. .addGap(18, 18, 18))) 150. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 151. .addGroup(layout.createSequentialGroup() 152. .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 58, javax.swing.GroupLayout.PREFERRED\_SIZE) 153. .addGap(55, 55, 55) 154. .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, 61, javax.swing.GroupLayout.PREFERRED\_SIZE)) 155. .addGroup(layout.createSequentialGroup() 156. .addGap(12, 12, 12) 157. .addComponent(jLabel5) 158. .addGap(89, 89, 89) 159. .addComponent(jLabel6)))) 160. .addGroup(layout.createSequentialGroup() 161. .addGap(31, 31, 31) 162. .addComponent(jLabel8))) 163. .addGap(29, 29, 29))) 164. .addContainerGap(40, Short.MAX\_VALUE)) 165. ); 166. layout.setVerticalGroup( 167. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 168. .addGroup(layout.createSequentialGroup() 169. .addGap(27, 27, 27) 170. .addComponent(jLabel1) 171. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE) 172. .addComponent(jLabel2) 173. .addComponent(jButton1) 174. .addComponent(jLabel3) 175. .addComponent(jLabel4)) 176. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED) 177. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING) 178. .addGroup(layout.createSequentialGroup() 179. .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 128, javax.swing.GroupLayout.PREFERRED\_SIZE) 180. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE) 181. .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 128, javax.swing.GroupLayout.PREFERRED\_SIZE) 182. .addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)) 183. .addGroup(layout.createSequentialGroup() 184. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE) 185. .addComponent(jLabel5) 186. .addComponent(jLabel6) 187. .addComponent(jLabel7)) 188. .addGap(1, 1, 1) 189. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE) 190. .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE) 191. .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE) 192. .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)) 193. .addGap(18, 18, 18) 194. .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE) 195. .addComponent(jButton2) 196. .addComponent(jButton3)) 197. .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 28, Short.MAX\_VALUE) 198. .addComponent(jLabel8) 199. .addGap(18, 18, 18) 200. .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED\_SIZE, 128, javax.swing.GroupLayout.PREFERRED\_SIZE) 201. .addGap(26, 26, 26)))) 202. ); 204. pack(); 205. }// </editor-fold> 207. **private** **void** jButton1ActionPerformed(java.awt.event.ActionEvent evt) { 208. //File Selection 209. JFileChooser jf = **new** JFileChooser(); 210. **int** a = jf.showOpenDialog(**null**); 211. System.out.println(a); 212. **if**(a == JFileChooser.APPROVE\_OPTION){ 213. FileReader bf = **null**; 214. **try** { 215. ff = jf.getSelectedFile(); 216. JOptionPane.showMessageDialog(rootPane, "Nombre del archivo \n" + ff.getName()); 217. //Lectura del archivo y escritura en la plantilla 218. bf = **new** FileReader(ff); 219. jTextArea2.read(bf, ""); 220. } **catch** (Exception ex) { 221. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 222. } **finally** { 223. **try** { 224. bf.close(); 225. } **catch** (IOException ex) { 226. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 227. } 228. } 229. } 230. } 232. **private** **void** jButton2ActionPerformed(java.awt.event.ActionEvent evt) { 233. String a,number; 234. **int** alpha,i,n; 235. number = jTextField3.getText(); 236. n = Integer.parseInt(number); 237. ArrayList<Integer> fact = **new** ArrayList<Integer>(); 238. fact = factors(n); 239. a = jTextField1.getText(); 240. alpha = Integer.parseInt(a); 241. alpha = multiInverse(alpha,n,fact); 242. i = serch(alpha); 243. **if**(i == -1){ 244. JOptionPane.showMessageDialog(rootPane, "Enter an other value of alpha\nThe value "+alpha+" is no accepted"); 245. } 246. **else** 247. encrypt(); 248. } 250. **private** **void** jButton3ActionPerformed(java.awt.event.ActionEvent evt) { 251. // Decryption method 252. String a; 253. **int** alpha,i; 254. a = jTextField1.getText(); 255. alpha = Integer.parseInt(a); 256. i = serch(alpha); 257. **if**(i == -1){ 258. JOptionPane.showMessageDialog(rootPane, "Enter an other value of alpha\nThe value "+alpha+" is no accepted"); 259. } 260. **else** 261. decrypt(); 263. } 265. **private** **void** jTextField3ActionPerformed(java.awt.event.ActionEvent evt) { 266. // TODO add your handling code here: 267. } 269. **private** **void** jTextField1ActionPerformed(java.awt.event.ActionEvent evt) { 270. // TODO add your handling code here: 271. } 273. /\*\* 274. \* @param args the command line arguments 275. \*/ 276. File ff; 277. String alphafet = "abcdefghijklmnopqrstuvwxyz"; 278. **static** **int**[] alpha1; 279. **public** **static** **void** main(String args[]) { 280. /\* Set the Nimbus look and feel \*/ 281. //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) "> 282. /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel. 283. \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html 284. \*/ 285. **try** { 286. **for** (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) { 287. **if** ("Nimbus".equals(info.getName())) { 288. javax.swing.UIManager.setLookAndFeel(info.getClassName()); 289. **break**; 290. } 291. } 292. } **catch** (ClassNotFoundException ex) { 293. java.util.logging.Logger.getLogger(AffineInterface.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex); 294. } **catch** (InstantiationException ex) { 295. java.util.logging.Logger.getLogger(AffineInterface.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex); 296. } **catch** (IllegalAccessException ex) { 297. java.util.logging.Logger.getLogger(AffineInterface.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex); 298. } **catch** (javax.swing.UnsupportedLookAndFeelException ex) { 299. java.util.logging.Logger.getLogger(AffineInterface.**class**.getName()).log(java.util.logging.Level.SEVERE, **null**, ex); 300. } 301. //</editor-fold> 303. /\* Create and display the form \*/ 304. java.awt.EventQueue.invokeLater(**new** Runnable() { 305. **public** **void** run() { 306. **new** AffineInterface().setVisible(**true**); 307. } 308. }); 309. } 310. **public** **static** ArrayList<Integer> factors(**int** n){ 311. System.out.println("Numero a descomponer en factores "+n); 312. ArrayList<Integer> factor = **new** ArrayList<Integer>(); 313. **int** d = 2; 314. **while**(n!= 1){ 315. **if**(n%d == 0){ 316. **if**(!factor.contains(d)){ 317. factor.add(d); 318. } 319. n = n/d; 320. } 321. **else**{ 322. d++; 323. } 324. } 325. System.out.println(factor); 326. **return** factor; 327. } 328. **public** **void** alphaNumbers(**int** n,ArrayList<Integer> factors){ 329. **int** i,j; 330. ArrayList<Integer> alphaca = **new** ArrayList<Integer>(); 331. System.out.println(factors.size()); 332. **for**(i = 0; i < n; i++){ 333. **for**(j = 0; j < factors.size(); j++){ 334. **if**((i%factors.get(j)) != 0){ 335. **if**(!alphaca.contains(i)) 336. alphaca.add(i); 337. } 338. j++; 339. } 340. } 341. alpha1 = **new** **int**[alphaca.size()]; 342. **for**(i = 0; i < alphaca.size(); i++) 343. alpha1[i] = alphaca.get(i); 344. } 345. **public** **int**  inverse(**int** num,**int** module){ 346. **int** number = num + module; 347. **while** (number < 0){ 348. number =number + module; 349. } 350. **return** number; 351. } 352. **public** **int** multiInverse(**int** num,**int** n,ArrayList<Integer> factors){ 353. alphaNumbers(n,factors); 354. System.out.println("Alpha = "+num); 355. **int** i = 0; 356. **for**(i = 0; i < alpha1.length; i++){ 357. System.out.print(alpha1[i]+" "); 358. } 359. **for**(i = 0; i < alpha1.length;i++){ 360. **if**((num \* alpha1[i])%n == 1){ 361. System.out.println(num +"\*"+"alpha1"+alpha1[i]); 362. num = alpha1[i]; 363. } 364. } 365. System.out.println("num: "+num); 366. **return** num; 367. } 368. **public** **int** gcd(**int** a, **int** n){ 369. /\*\* 370. \* This validation help to calculate the greatest common divisor of two 371. \* numbers using the Eclides Algorithm 372. \*/ 373. **int** t; 374. **while**(a > 0){ 375. t = a; 376. a = n % a; 377. n = t; 378. } 379. **return** n; 380. } 381. **public** **static** **long**[] extended\_euclides(**long** a, **long** n){ 382. **long** x,y; 383. **long** result[] = **new** **long**[2]; 384. **if** (n == 0){ 385. a = a; 386. result[0] = 0; 387. result[1] = 0; 388. } **else**{ 389. **long** x2 = 1, x1 = 0, y2 = 0, y1 = 1; 390. **long** q = 0, r = 0; 391. **while**(n>0) 392. { 393. q = (a/n); 394. r = a - q\*n; 395. x = x2-q\*x1; 396. y = y2 - q\*y1; 397. a = n; 398. n = r; 399. x2 = x1; 400. x1 = x; 401. y2 = y1; 402. y1 = y; 403. } 404. result[0] = x2; 405. result[1] = y2; 406. } 407. **return** result; 408. } 409. **public** **void** encrypt(){ 410. PrintWriter EncryptOut = **null**; 411. **try** { 412. String a, b,line,nAlphabet; 413. //Funtion to write in the FILE 414. EncryptOut = **new** PrintWriter("out.txt"); 415. **int** alpha, beta,n; 416. a = jTextField1.getText(); 417. b = jTextField2.getText(); 418. nAlphabet = jTextField3.getText(); 419. **if**(a.matches("[0-9]+") && b.matches("[0-9]+") && nAlphabet.matches("[0-9]+")){ 420. alpha = Integer.parseInt(a); 421. beta = Integer.parseInt(b); 422. n = Integer.parseInt(nAlphabet); 423. **if**(gcd(alpha,n) != 1){ 424. JOptionPane.showMessageDialog(rootPane, "Enter a diferent value of alpha or N"); 425. }**else**{ 426. **long** r[] = **new** **long**[1]; 427. r = extended\_euclides(alpha,n); 428. jTextArea3.append("Encryption\n C = "+alpha+"p"+" + "+beta +" mod "+n+"\n"); 429. } 430. FileReader lectura = **null**; 431. **try** { 432. JOptionPane.showMessageDialog(rootPane, "The entered value is \nalpha = "+alpha+"\nbeta =  "+beta); 433. lectura = **new** FileReader(ff); 434. BufferedReader buffer = **new** BufferedReader(lectura); 435. StringBuilder cadena = **new** StringBuilder(); 436. **int** i,p; 437. **while**((line = buffer.readLine()) != **null**){ 438. //here where the algorithm takes action 439. line = line.toLowerCase(); 440. **for**(i = 0; i < line.length(); i++){ 441. p = alphafet.indexOf(line.charAt(i)); 442. **if**(p != -1){ 443. cadena.append(alphafet.charAt(((alpha)\*(p) + beta)%26)); 444. } 445. } 446. System.out.println(cadena.toString().toUpperCase()); 447. EncryptOut.write(cadena.toString().toUpperCase()); 448. //salidaCifrada.println(); 449. cadena.setLength(0); 450. } 451. EncryptOut.flush(); 452. } **catch** (FileNotFoundException ex) { 453. JOptionPane.showMessageDialog(rootPane,"File not found"); 454. } **catch** (IOException ex) { 455. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 456. } **finally** { 457. **try** { 458. lectura.close(); 459. } **catch** (IOException ex) { 460. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 461. } 462. } 463. } 464. **else**{ 465. JOptionPane.showMessageDialog(rootPane, "Invalid data, please enter numeric values"); 466. } 467. } **catch** (FileNotFoundException ex) { 468. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 469. } **finally** { 470. EncryptOut.close(); 471. } 472. ff = **new** File("out.txt"); 473. //Write in the jTextArea1 474. **if**(ff.getPath() != **null**){ 475. **try** { 476. FileReader bf = **null**; 477. bf = **new** FileReader(ff); 478. **try** { 479. jTextArea1.read(bf, ""); 480. } **catch** (IOException ex) { 481. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 482. }**finally**{ 483. **try** { 484. bf.close(); 485. } **catch** (IOException ex) { 486. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 487. } 488. } 490. } **catch** (FileNotFoundException ex) { 491. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 492. } 493. } 494. } 495. **public** **void** decrypt(){ 496. System.out.println(ff.getPath()); 497. PrintWriter EncryptOut = **null**; 498. **try** { 499. String a, b,line,nAlphabet; 500. //Funtion to write in the FILE 501. EncryptOut = **new** PrintWriter("outDecrypt.txt"); 502. **int** alpha, beta,n,index; 503. a = jTextField1.getText(); 504. b = jTextField2.getText(); 505. nAlphabet = jTextField3.getText(); 506. **if**(a.matches("[0-9]+") && b.matches("[0-9]+") & nAlphabet.matches("[0-9]+")){ 507. FileReader lectura = **null**; 508. **try** { 509. alpha = Integer.parseInt(a); 510. beta = Integer.parseInt(b); 511. n = Integer.parseInt(nAlphabet); 512. **if**(gcd(alpha,n) != 1){ 513. JOptionPane.showMessageDialog(rootPane, "Enter a diferent value of alpha or N"); 514. }**else**{ 515. **long** r[] = **new** **long**[1]; 516. r = extended\_euclides(alpha,n); 517. } 518. JOptionPane.showMessageDialog(rootPane, "The entered value is \nalpha = "+alpha+"\nbeta =  "+beta); 519. //INVERSO ADITIVO 520. beta = inverse(-beta,n); 521. //INVERSO MULTIPLICATIVO 522. System.out.println("Inverso Multi"); 523. index = serch(alpha); 524. ArrayList<Integer> factors = **new** ArrayList<Integer>(); 525. factors = factors(n); 526. System.out.println("FACTTTORS alpha = "+alpha+"\nn = "+n); 527. System.out.println(factors); 528. alpha = multiInverse(alpha,n,factors); 529. beta = (alpha\*beta)%n; 530. jTextArea3.append("Decryption\n p = "+alpha+"p"+" + "+beta +" mod "+n+"\n"); 531. lectura = **new** FileReader(ff); 532. BufferedReader buffer = **new** BufferedReader(lectura); 533. StringBuilder cadena = **new** StringBuilder(); 534. **int** i,p; 535. **while**((line = buffer.readLine()) != **null**){ 536. //here where the algorithm takes action 537. line = line.toLowerCase(); 538. **for**(i = 0; i < line.length(); i++){ 539. p = alphafet.indexOf(line.charAt(i)); 540. **if**(p != -1){ 541. cadena.append(alphafet.charAt(((alpha)\*((p) + beta))%26)); 542. } 543. } 544. System.out.println(cadena.toString().toLowerCase()); 545. EncryptOut.write(cadena.toString().toLowerCase()); 546. //salidaCifrada.println(); 547. cadena.setLength(0); 548. } 549. EncryptOut.flush(); 550. } **catch** (FileNotFoundException ex) { 551. JOptionPane.showMessageDialog(rootPane,"File not found"); 552. } **catch** (IOException ex) { 553. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 554. } **finally** { 555. **try** { 556. lectura.close(); 557. } **catch** (IOException ex) { 558. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 559. } 560. } 561. } 562. **else**{ 563. JOptionPane.showMessageDialog(rootPane, "Invalid data, please enter numeric values"); 564. } 565. } **catch** (FileNotFoundException ex) { 566. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 567. } **finally** { 568. EncryptOut.close(); 569. } 570. ff = **new** File("outDecrypt.txt"); 571. //Write in the jTextArea1 572. **if**(ff.getPath() != **null**){ 573. **try** { 574. FileReader bf = **null**; 575. bf = **new** FileReader(ff); 576. **try** { 577. jTextArea1.read(bf, ""); 578. } **catch** (IOException ex) { 579. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 580. }**finally**{ 581. **try** { 582. bf.close(); 583. } **catch** (IOException ex) { 584. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 585. } 586. } 588. } **catch** (FileNotFoundException ex) { 589. Logger.getLogger(AffineInterface.**class**.getName()).log(Level.SEVERE, **null**, ex); 590. } 591. } 592. } 593. **public** **int** serch(**int** alpha){ 594. **int** index = -1,i; 595. **for**(i = 0; i < alpha1.length; i++){ 596. **if**(alpha1[i] == alpha){ 597. index = i; 598. } 599. } 600. **return** index; 601. } 602. **public** **static** **int** phi(**int** n){ 603. ArrayList<Integer> factors = **new** ArrayList<Integer>(); 604. **int** result = n; 605. **int** p; 606. // Consider all prime factors of n and for every prime 607. // factor p, multiply result with (1 - 1/p) 608. **for** (p = 2; p \* p <= n;p++) { 610. // Check if p is a prime factor. 611. **if** (n % p == 0) { 612. // If yes, then update n and result 613. **while** (n % p == 0) 614. n /= p; 615. result -= result / p; 616. } 617. } 619. // If n has a prime factor greater than sqrt(n) 620. // (There can be at-most one such prime factor) 621. **if** (n > 1) 622. result -= result / n; 623. **return** result; 624. } |

<http://www.planetb.ca/syntax-highlight-word>

\*About figures or tables\*

Using figures such as diagrams, tables, graphs, charts or maps can be a very useful way to show and emphasise information in your report.

Figures essential to the report should be smoothly and correctly integrated and should be explained and referred to in the main body of the report.

Example:

