Alumno:	García González Aarón Antonio
Grupo:	3CM3
Unidad de Aprendizaje:	Crptography
Profesora:	Cortez Duarte Nidia Asunción
Practica #1:	Algoritmo Vigene, Affine y de Euclides extendido.
Fecha:	Viernes 30 de octubre de 2020.

Resumen

A lo largo de esta primera práctica se realizó la codificación de los algoritmos de cifrado y descifrado Vigenere y Affine vistos en las sesiones teóricas previamente, en el caso de Affine se utiliza el algoritmo de Euclides extendido para verificar que α y la longitud del alfabeto son números coprimos y, por ende, existirá un inverso multiplicativo (lógica modular) de α y así un correcto descifrado.

Se utiliza el código ASCII como alfabeto base, donde el usuario puede seleccionar un número menor o igual que 256 para la longitud del alfabeto que desea cubrir, siempre considerando que dicho número seleccionado cubra todos los caracteres utilizados en la llave para el caso del algoritmo Vigenere, de lo contrario se observa que hay errores.

Índice

1. Probar AEE

Dado que la interfaz de usuario solicitada esta en el nivel de abstracción mas alto, no incluye la validación directa de los parámetros α y n, por lo que exclusivamente esta parte la realizaremos desde consola. Probar con:

- α = 5, n = 30

```
int α = 5;
int n = 20;
int mcd = AritmeticaModular.mcd(α, n)[0];
int inverso_multiplicativo = AritmeticaModular.mcd(α, n)[1];

if (mcd != 1){
   System.out.println("mcd(" + α + ", " + n + ") = " + mcd);
   System.out.println("Dado lo anterior, " + α + ", no tiene inverso multiplicativo en " + n);
}else{
   System.out.println("mcd(" + α + ", " + n + ") = " + mcd);
   System.out.println("Inverso multiplicativo de " + α + " = " + inverso_multiplicativo);
}
```

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(5, 20) = 5
Dado lo anterior, 5, no tiene inverso multiplicativo en 20

Process finished with exit code 0
```

- α = 97, n = 239

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(97, 239) = 1
Inverso multiplicativo de 97 = 69

Process finished with exit code 0
```

- α = 11111, n = 12345

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(11111, 12345) = 1
Inverso multiplicativo de 11111 = 2471

Process finished with exit code 0
```

- a= 13, n = 99991

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(13, 99991) = 1
Inverso multiplicativo de 13 = -38458

Process finished with exit code 0
```

- a = 10009, n = 104729

```
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
mcd(100009, 104729) = 1
Inverso multiplicativo de 100009 = -44177

Process finished with exit code 0
```

2. Aritmética modular (Códigos)

a) Inverso aditivo

```
public static int InversoAditivo(int numero, int longitud_de_alfabeto)
    {
        if(numero > 0) {
            return (longitud_de_alfabeto - numero);
        }
        else {
            return longitud_de_alfabeto + numero;
        }
    }
```

a) Máximo común divisor

```
1.  /* Regresa array[d,a,b] donde, d = gcd(p,q), ap + bq = d
2.          donde a, de ser negativo no calcula el inverso aditivo.
3.      */
4.      public static int[] mcd(int p, int q) {
5.          if(q==0) {
6.              return new int[] {p,1,0};
7.          }
8.          int [] vals = mcd(q, p%q);
9.          int d = vals[0];
10.         int a = vals[2];
11.         int b = vals[1] - (p/q) * vals[2];
12.
13.         return new int []{d,a,b};
14.     }
```

b) Inverso multiplicativo

```
1.  public static int InversoMultiplicativo(int p, int n)
2.      {
3.          int inverso_multiplicativo = mcd(p, n)[1];
4.
5.          if(inverso_multiplicativo<0)
6.              return InversoAditivo(inverso_multiplicativo, n);
7.          else
8.              return inverso_multiplicativo;
9.      }
```

3. Algoritmo de Euclides y Euclides extendido (Explicación)

El algoritmo de Euclides es un método eficiente de calcular el máximo común divisor entre dos números, este algoritmo no es muy eficiente de implementar en computadora, ya que se tienen que memorizar todos los valores de residuo.

Algoritmo 1 de Euclides

Entrada: Valores a y b pertenecientes a un dominio euclídeo

Salida: Un máximo común divisor de a y b

1. $r_0 \leftarrow a, r_1 \leftarrow b$
2. $i \leftarrow 1$
3. **Mientras** $r_i \neq 0$ **haga lo siguiente:**
    1. $r_{i+1} \leftarrow r_{i-1} \mod r_i$
    2. $i \leftarrow i + 1$
4. **El resultado es:** r_{i-1}

Por otro lado, el Algoritmo de Euclides extendido, es una ligera modificación que permite además expresar al máximo común divisor como una combinación lineal, mediante las combinaciones lineales, al despejar los residuos nos permite obtener el inverso multiplicativo, y al mismo tiempo conocer el mcd.

Algoritmo 2 de Euclides extendido

Entrada: Valores a y b pertenecientes a un dominio euclídeo

Salida: Un máximo común divisor de a y b, y valores s y t tales que $\mathrm{mcd}(a, b) = as + bt$

1. $r_0 \leftarrow a, r_1 \leftarrow b, s_0 \leftarrow 1, t_0 \leftarrow 0, s_1 \leftarrow 0, t_1 \leftarrow 1$
2. $i \leftarrow 1$
3. **Mientras** $r_i \neq 0$ **haga lo siguiente:**
    1. Divida r_{i-1} entre r_i para obtener el cociente q_i y el residuo r_{i+1}
    2. $s_{i+1} \leftarrow s_{i-1} - q_i s_i$
    3. $t_{i+1} \leftarrow t_{i-1} - q_i t_i$
    4. $i \leftarrow i + 1$
4. **El resultado es:** r_{i-1} es un máximo común divisor de a y b y se expresa $r_{i-1} = as_{i-1} + bt_{i-1}$

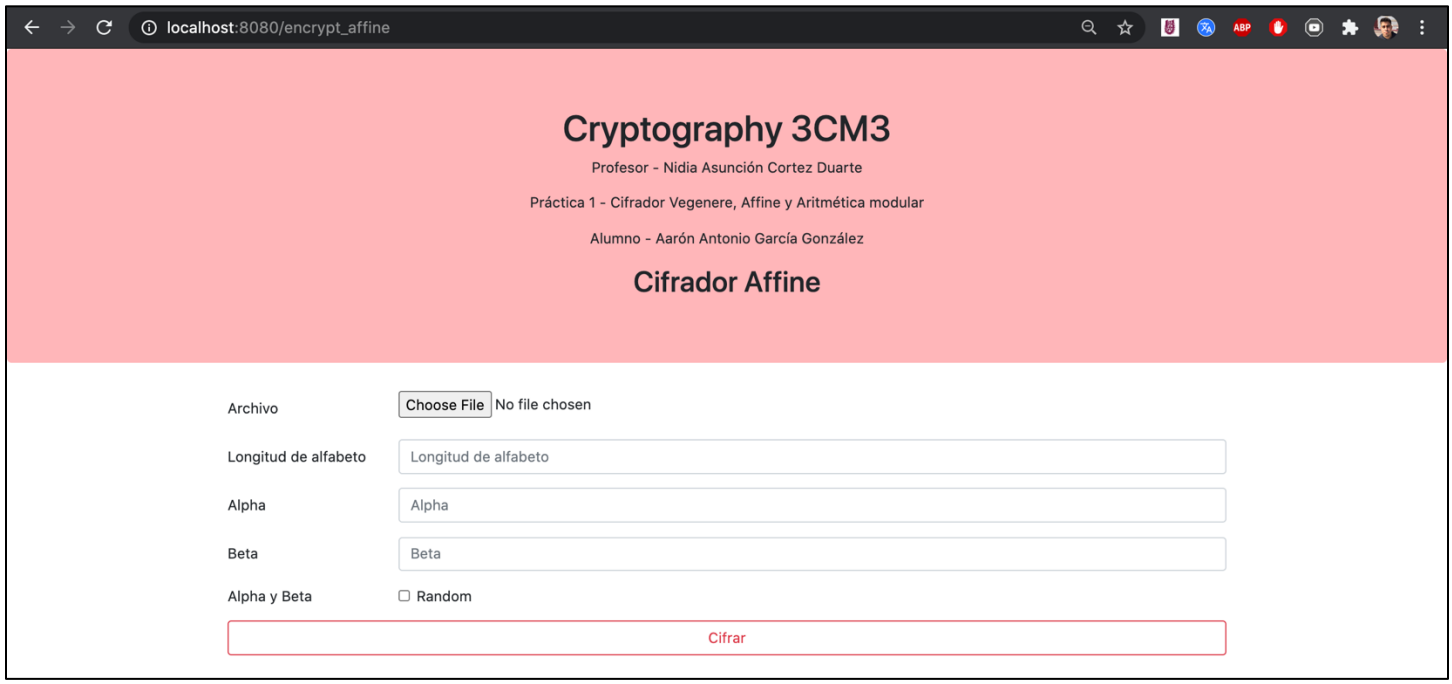
4. Interfaces de selección

a) Index

b) Vigenere cifrado

a) Vigenere descifrado

c) Affine cifrado



d) Affine descifrado

5. Cifrador Vigenere

a) Explicación

Este es un algoritmo de sustitución poli alfabético, esto debido a que una misma letra es posible mapearse o cifrarse a más de una en común. El funcionamiento de este algoritmo es muy sencillo, para cifrar lo que se tiene que hacer es:

1. Calcular la longitud del texto en claro
2. Repetir la llave hasta hacer coincidir con la longitud del texto en claro
3. Comparar posición a posición de cada carácter tanto del texto en claro como de la llave obtenida en el paso #2, y sumar los valores
4. A cada número obtenido en #3, aplicar modulo longitud del alfabeto
5. Mapear cada número al valor correspondiente con el alfabeto

Para descifrar, lo que se hace es:

6. Calcular el inverso aditivo de cada carácter de la llave
7. Con base en la llave inverso aditivo obtenida en #6, repetir la llave hasta hacer coincidir con la longitud del texto cifrado
8. Comparar posición a posición de cada carácter tanto del texto cifrado como de la llave obtenida en el paso #7, y sumar los valores
9. A cada número obtenido en #8, aplicar modulo longitud del alfabeto
10. Mapear cada número al valor correspondiente con el alfabeto

b) Código cifrador

```
/* aplicando la formula de cifrado: C_n = P_n + K mod longitud_alfabeto, donde:
            C_n, es el carácter del texto cifrado resultante
            P_n, es la letra de la posicion n del texto en claro
            K, es el valor del carácter de la llave
            longitud_alfabeto, es el tamanio del alfabeto
     */
    public static String Encrypt(String plain_text, String key, int longitud_alfabeto)
    {
        StringBuilder cipher_text = new StringBuilder();

        int counter_key_value = 0;
        for (int i=0; i<plain_text.length(); i++)
        {
            if(counter_key_value >= key.length()){
                counter_key_value = 0;
            }

            int P_n = (int)plain_text.charAt(i);
            int K = (int)key.charAt(counter_key_value);
            int C_n;

            if(longitud_alfabeto == 26){ // alfabeto ingles
                if(P_n < 97 || P_n > 122){
                    C_n = P_n;
                }else{
                    C_n = ( ( P_n + K - 194 ) % longitud_alfabeto ) + 97;
                }
            }else{
                C_n = (P_n + K) % longitud_alfabeto;
            }

            cipher_text.append((char)C_n);

            counter_key_value += 1;
        }

        return cipher_text.toString();
    }
```

c) Código descifrador

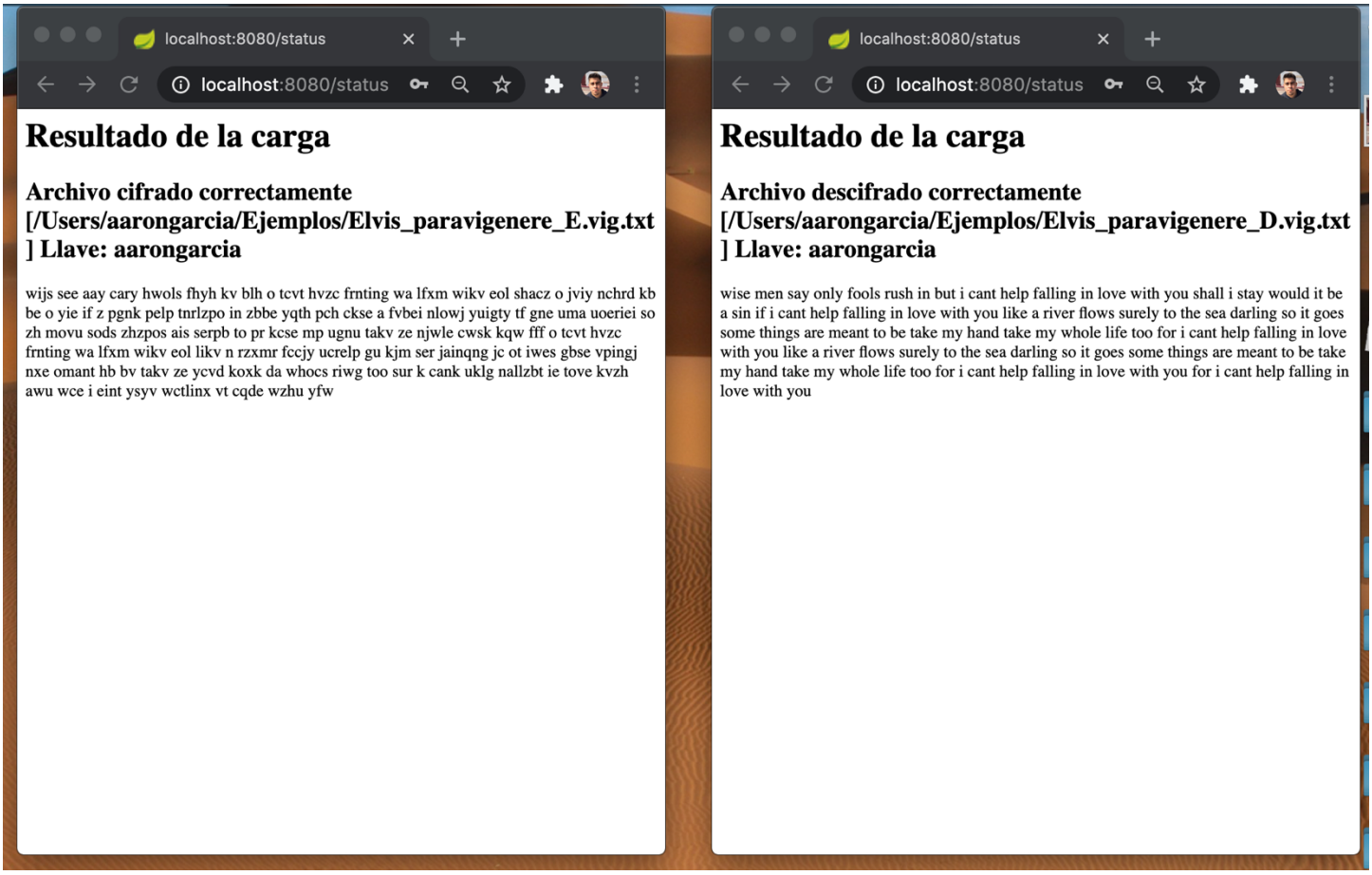
```
/* aplicando la formula de des cifrado: P_n = C_n + (-K) mod longitud_alfabeto, donde:
     P_n, es la letra del texto claro resultante
     C_n, es la letra de la posicion n del texto cifrado
     -K, es el inverso aditivo del carácter de la llave
     longitud_alfabeto, es el tamanio del alfabeto

 public static String Decrypt(String cipher_text, String key, int longitud_alfabeto)
 {
     StringBuilder decipher_text = new StringBuilder();

     int counter_key_value = 0;
     for (int i=0; i<cipher_text.length(); i++)
     {
         if(counter_key_value >= key.length()){
             counter_key_value = 0;
         }
```

```
17.
18.          int C_n = (int)cipher_text.charAt(i);
19.          int K = AritmeticaModular.InversoAditivo((int)key.charAt(counter_key_value), longitud_alfabe
    to);
20.          int P_n;
21.
22.          if(longitud_alfabeto == 26){
23.              if(C_n < 97 || C_n > 122){
24.                  P_n = C_n;
25.              } else{
26.                  P_n = ( ( C_n + K) % longitud_alfabeto ) + 97;
27.              }
28.          }else{
29.              P_n = (C_n + K) % longitud_alfabeto;
30.          }
31.
32.          decipher_text.append((char)P_n);
33.
34.          counter_key_value += 1;
35.      }
36.
37.      return decipher_text.toString();
38.  }
```

d) Prueba



6. Cifrador Affine

a) Explicación

Este es algoritmo de sustitución mono-alfabética, esto debido a que todos los caracteres iguales de un texto se mapean a otro mismo con base en las funciones de cifrado y descifrado, este algoritmo es muy sencillo, la manera en que se puede cifrar es:

1. Verificar que α y la longitud del alfabeto son coprimos, de serlo, el algoritmo puede continuar
2. A cada valor según la posición de cada carácter del texto en claro se multiplicará por el valor α.
3. A cada valor obtenido en #2, se le sumara el valor de β
4. A cada valor obtenido en #3, se le aplicara modulo longitud del alfabeto
5. Mapear cada número al valor correspondiente con el alfabeto

Para descifrar, lo que se debe hacer es:

6. Calcular el inverso multiplicativo de α
7. Calcular el inverso aditivo de β
8. A cada valor según la posición de cada carácter del texto cifrado se le sumará el valor de β obtenido en #7
9. A cada valor de #8 se multiplicará por el valor obtenido de α en #6
10. A cada valor obtenido en #9, se le aplicara modulo longitud del alfabeto
11. Mapear cada número al valor correspondiente con el alfabeto

b) Código cifrador

```
1.  /*
2.      * Funcion que cifra con base a una llave alpha valida
3.      *
4.      *  aplicando la formula de cifrado: C_n = alpha * P_n + beta mod longitud_del_alfabeto, donde:
5.      *  C_n, es el carácter del texto cifrado resultante
6.      *  P_n, es la letra de la posicion n del texto en claro
7.      *  alpha, es el valor multiplicativo de la llave
8.      *  beta, es el valor aditivo de la llave
9.      * */
10.     public static String Encrypt(String plain_text, int alphabet_length, int alpha, int beta)
11.     {
12.         StringBuilder cipher_text = new StringBuilder();
13.
14.         for (int i=0; i<plain_text.length(); i++)
15.         {
16.             int P_n = (int)plain_text.charAt(i);
17.             int C_n = ((alpha * P_n) + beta) % alphabet_length;
18.             cipher_text.append((char)C_n);
19.         }
20.
21.         return cipher_text.toString();
22.     }
```

c) Código descifrador

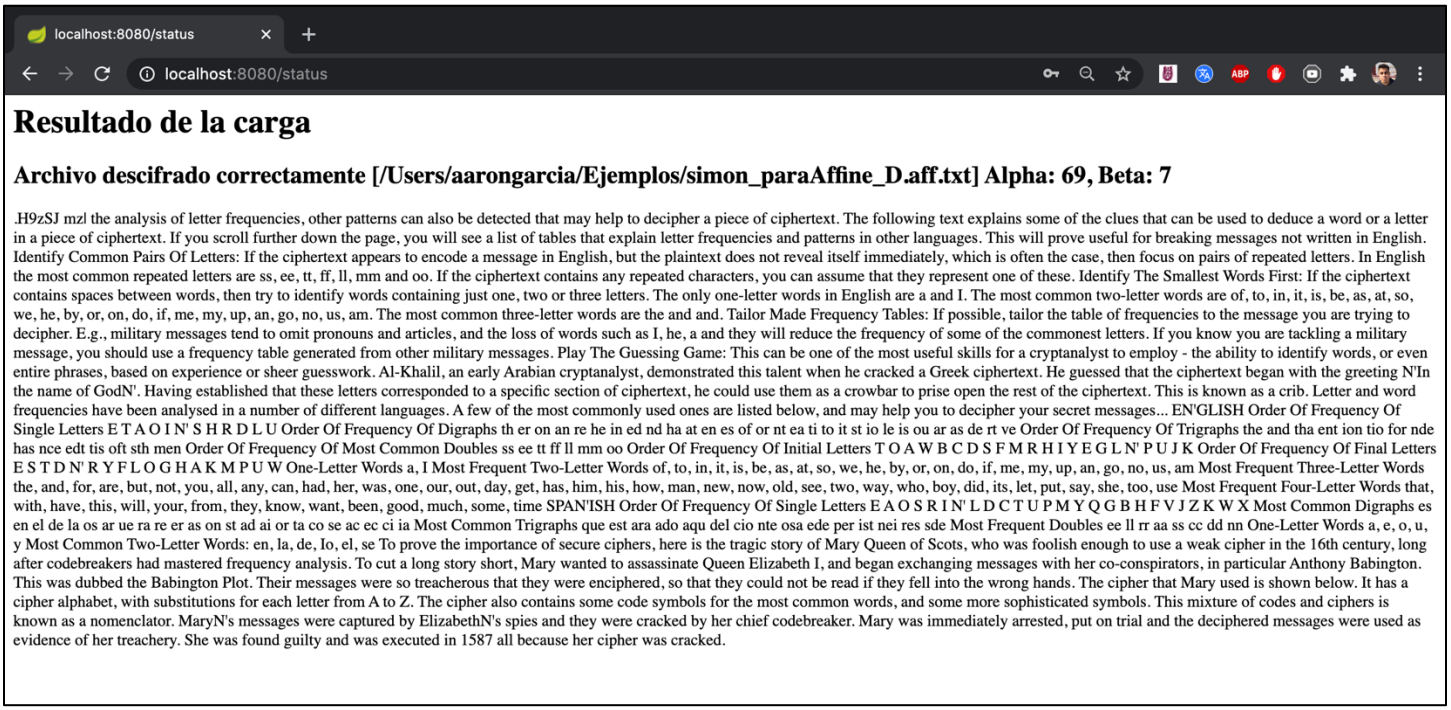
```
1.  /*
2.   *  Aplicando la formula de des cifrado: P_n = alpha^-1 * [ C_n + (-
    beta) ] mod longitud_del_alfabeto, donde:
3.   *  C_n, es el carácter del texto en claro descifrado
4.   *  P_n, es la letra de la posicion n del texto cifrado
5.   *  alpha, es el valor multiplicativo de la llave en inverso multiplicativo
6.   *  beta, es el valor aditivo de la llave en inverso aditivo
7.   * */
8.  public static String Decrypt(String cipher_text, int alphabet_length, int alpha, int beta)
9.  {
10.     StringBuilder decipher_text = new StringBuilder();
11.
12.     for (int i=0; i<cipher_text.length(); i++)
13.     {
14.         int C_n = (int)cipher_text.charAt(i);
15.         int alpha_inverso_multiplicativo = AritmeticaModular.InversoMultiplicativo(alpha, alphabet_le
    ngth);
16.         int beta_inverso_aditivo = AritmeticaModular.InversoAditivo(beta, alphabet_length);
17.         int P_n = ( alpha_inverso_multiplicativo * ( C_n + beta_inverso_aditivo ) ) % alphabet_length
    ;
18.
19.         decipher_text.append((char)P_n);
20.     }
21.     return decipher_text.toString();
22. }
```

d) Pruebas

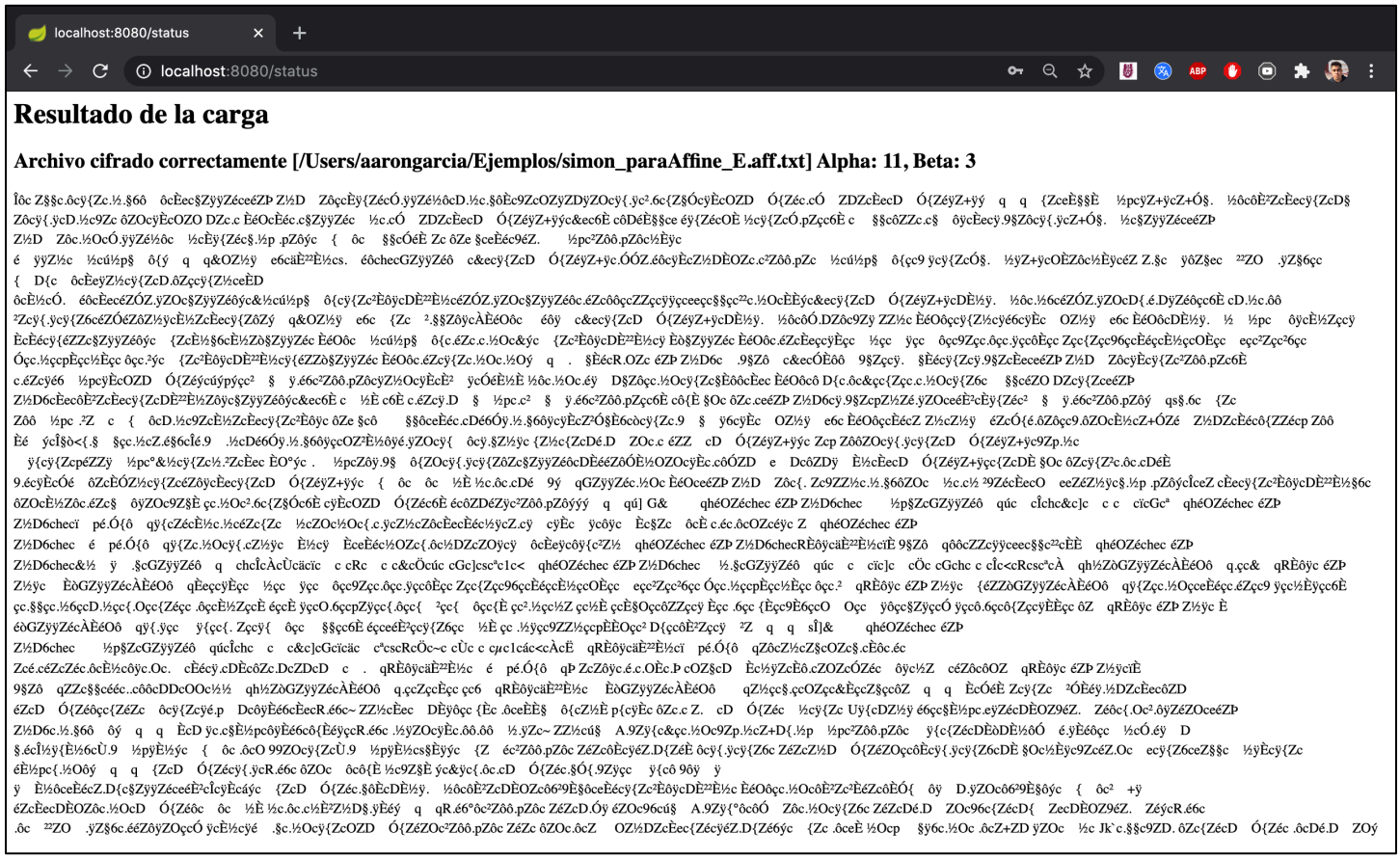
i. α = 20, Z = 256, β = 3

ii. α = 69, Z = 128, β = 7

Modificar los 10 primeros caracteres del texto cifrado y descifrar



i. $\alpha = 11, Z = 256, \beta = 5$



Modificar los 10 primeros caracteres del texto cifrado y descifrar

7. Conclusiones

En lo personal si tuve algunas dificultades al implementar el algoritmo de Vigenere, ya que la especificación de la practica acoto a utilizar el alfabeto ingles, mis programas funcionan a partir del código ASCII, mi cifrado/descifrado Vigenere funciona en todo el alfabeto ASCII, por lo que añadí una condición para verificar cuando la longitud del alfabeto sea igual a 26, tratarlo de manera diferente, y también dado que permite elegir el tamaño del alfabeto, hay que tener cuidado que la clave este dentro de los primeros n caracteres que especifico el usuario, de lo contrario dará errores, para el cifrado Affine, dado que el alfabeto es mas general, todo fue mas sencillo, es decir a la expresión original para descifrar no le modifique nada
Para mi gusto ambos cifradores tienen lo suyo, creo que es una excelente aproximación a lo que va de esta unidad de aprendizaje, también tuve que investigar algunas cosas adicionales, dado que la parte de la interfaz grafica la hago en web junto con Spring, meter un archivo a un formulario junto con otros datos y casillas de verificación en conjunto, lo desconocía.

8. Tabla de cotejo

1. Documento		**tú**
La portada tiene: nombre completo, materia, nombre de profesor, fecha, logotipos, título de práctica y un resumen. (ver ejemplo anexo)	1	1
En media cuartilla con tus palabras explicar el AE y AEE, qué es y para qué sirve cada uno de ellos.	1	1
Código correspondiente a las dos funciones AE y AEE y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Media cuartilla con tus palabras sobre el algoritmo Affine	1	
Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Media cuartilla con tus palabras sobre el algoritmo Vigenère	1	1
Código correspondiente a las dos funciones (cifrado y descifrado) así como el cálculo de la llave de descifrado y captura de pantalla de las ejecuciones de las pruebas solicitadas.	2	2
Conclusiones en donde se exprese principalmente las dificultades de la implementación de la práctica (si es que las hubo) así como una reflexión sobre la diferencia entre cifradores de sustitución mono alfabética y poli alfabético.	1	1
El código tiene formato (sugiero utilizar http://www.planetb.ca/syntax-highlight-word)	1	1
Todas las imágenes en el documento tienen título y se referencian en alguna parte del mismo. (Ej. “en la imagen 1 se muestra …”)	1	0
Programa		
El programa cuenta con interfaz gráfica que le permita al usuario elegir la opción deseada: algoritmo y cifrado o descifrado y que reciba los parámetros (en caso de ser necesario)	2	2
La interfaz permite seleccionar el archivo que se va a cifrar/descifrar	2	2
Su programa genera el archivo cifrado con el mismo nombre del archivo de entrada más la extensión .vig o .aff	2	2
Las funciones AE y AEE mandan mensaje “prueba con otro valor” en caso de que alpha no sea coprimo con n	1	1
TOTAL	20	**19**

García González Aarón Antonio