TABLES

Table: Cryptographic Methods

Field name	**Input (indirectly from the user) Type**	**Return Type**	**Description**
inMess	Text	None	Contains the string of characters that the user desires to encipher or decipher.
inStrucMorse	None	Text	Displays the instructions regarding how to input a functional Morse code
inStrucCae	None	Text	Displays the instructions regarding how to input a functional Caesar Cipher code
inStrucZig	None	Text	Displays the instructions regarding how to input a functional ZigZag code
inStrucVig	None	Text	Displays the instructions regarding how to input a functional Vigenère code
inStrucTimeRel	None	Text	Displays the instructions regarding how to input a functional Timed Release code
Encrypt	Text (includes specialized characters) and Key	Text	Method that returns an encrypted output of the initial input from user by using Key (keyEn).
Decrypt	Text(includes specialized characters) and Key	Text	Method that returns a decrypted output of the initial input from user by using a Key (keyDe).
morseEn	‘.’ Character and ‘-‘ Character	Text (alphabetic and numeric)	A method key that is used to encrypt input from the user into Morse code.
morseDe	Text (alphabetic and numeric)	‘.’ Character and ‘-‘ Character	A method key that is used to decrypt input from the user into Morse code.
caeSarEn	Alphabetic characters	Alphabetic characters	A method key that is used to encrypt input from the user using a Caesar Shift Cipher.
caeSarDe	Alphabetic characters	Alphabetic characters	A method key that is used to decrypt input from the user using a

			Caesar Shift Cipher decryption algorithm.
vigEn	Alphabetic characters.	Alphabetic characters (Functions by using multiple caeSar keys)	A method key that is used to encrypt input from the user by passing it through multiple caeSarEn methods to produce a message.
vigDe	Alphabetic characters enciphered	Alphabetic characters.	A method key that is used to decrypt input from the user by passing it through multiple caeSarDe methods to produce a message.
zigEn	Text (includes alphabetic, numeric, and special characters)	Text	A method key that is used to encrypt input from the user by using a sorting algorithm.
zigDe	Text	Text	A method key that I used to decrypt input from the user by using a resorting algorithm.

Table: Macros

Macro Name	**Description**
macroAuto	Runs LoginScreen in order to display a login screen
macroMen	Main menu that starts up after logging in.

Table: Login

Field name	**Input (indirectly from the user) Type**	**Output Type**	**Description**
LoginScreen	None	Textbox object for prompting for USER NAME and PASSWORD/ Displaying Program name/	A method that displays the initial login screen for the program
userN	Text	None	Variable that holds the input for the proposed request of acquiring the user name.
passWd	Text	None	Variable that holds the input for the proposed request of acquiring the user password.

COMPUSER	Text	None	Stores set user name remains constant until changed.
COMPPASS	Text	None	Stores set user password remains constant until changed.

Table: Main menu objects/Forms

Object Name	**Description**
textBox1	A textbox into which the user inputs the text they wish to manipulate.
textBox2	Displays the manipulated input of the user. Either an encrypted or decrypted message.
cryptoDec	A drop down menu that links to the encrypt and decrypt methods.
cryptoCrypt	A drop down menu that contains the desired type of cipher. Accesses the Cipher class.
cryptoKeys	Displays a key (not to be confused with method key) , for use with the Caesar Shift and Vigenère methods only, in a drop-down menu. Accesses the Key class.
outMess	Displays a button that initiates either the Encrypt or Decrypt method.

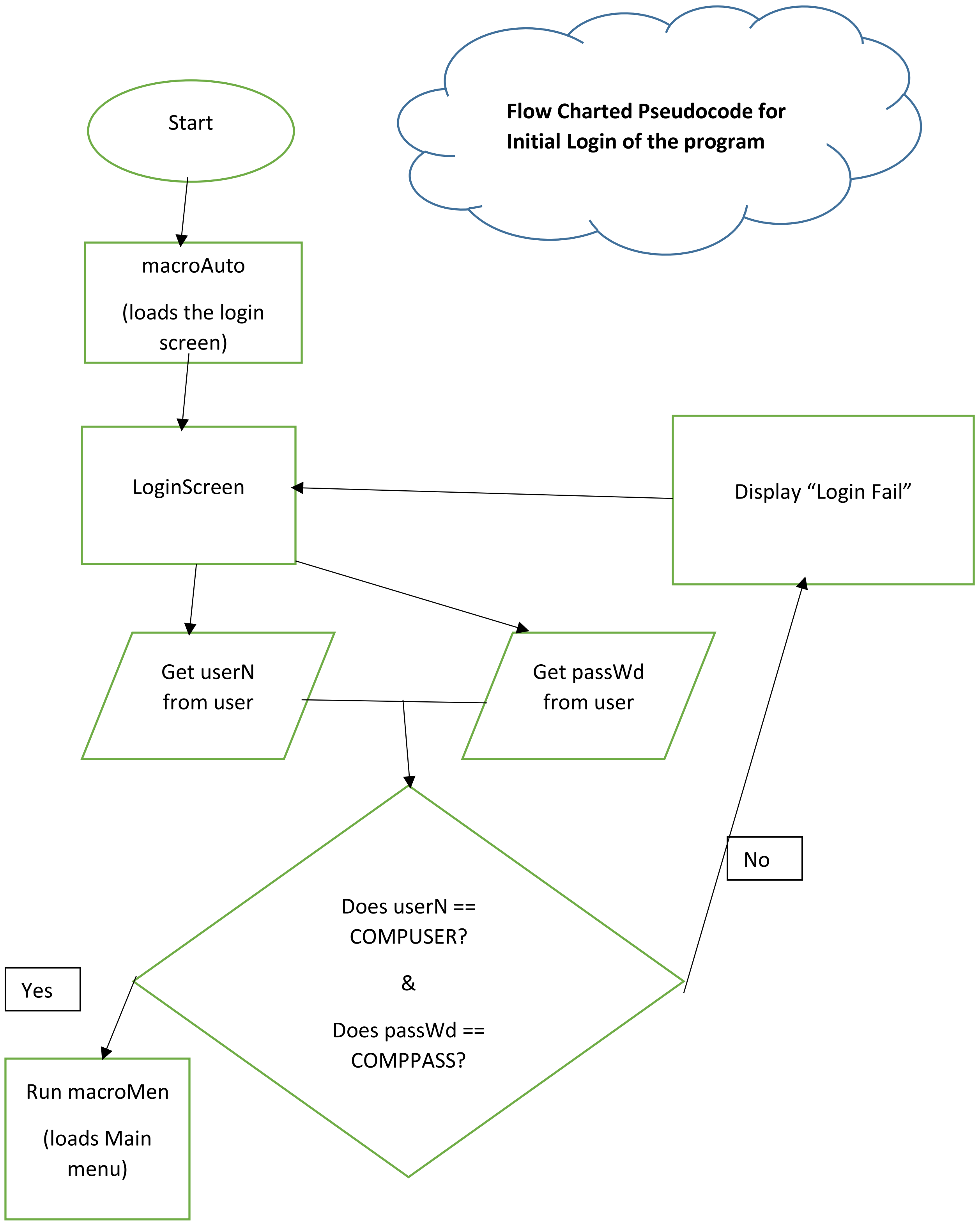
Table: Classes

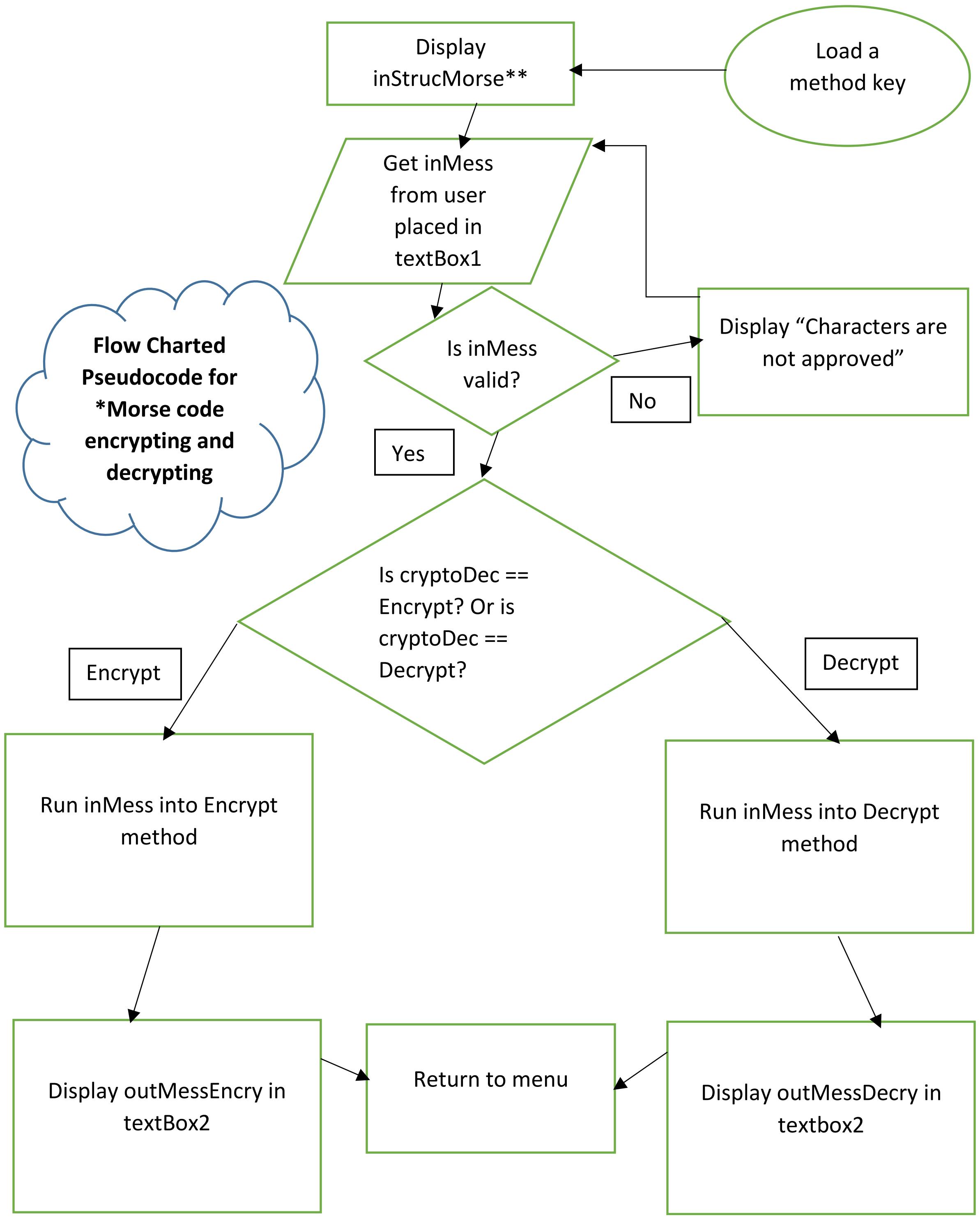
Class Name	**Description**
CaesarShift	Contains the different cipher methods for performing a Caesar Cipher.
CryptoVault	Contains the GUI form for the main menu.
LoginForm	Contains the GUI form for the login screen.
VigenereCipher	Contains the different cipher methods for performing a Vigenere Cipher.
ZigZag	Contains the different cipher methods for performing a ZigZag Cipher.
Morse	Contains the different cipher methods for Morse Code.

Table: Testing Plan

Test Expectations	**Test Execution**	**Example**	**Success Criterion Met**
Success of logging in loads the menu properly.	Displays objects located on the table titled *Main menu objects*	User enters his login and the function buttons, drop-down menus, and textbox are displayed in a neat and organized fashion as	A personal login (1) and a neat and presentable GUI (6).

		requested by the client, after the login is successful.	
Forms work	All five encryption methods and five decryption methods, which the user desires, link correctly to the buttons and drop-down menus provided. The output on the textbox is clear and contains no truncations from the original message.	Enter necessary protocols to either encrypt his message or decipher it and produces the desired message.	5 types of encryption listed on a drop-down menu from which to choose from (2), an option for encrypting and decrypting (3), textbox which to enter the message in order to be encrypted or decrypted (4), and a textbox to display the message either encrypted or decrypted (5).
GUI is pleasant	The login screen has the buttons and textbox separated in a proportional manner. Drop-down menu is on the left hemisphere of the workspace, and the two textbox are on the right hemisphere of the workspace.	Login in to program in order to view the workspace. The drop-down menu does not overlap or interfere with the textboxes.	A neat and presentable GUI (6) and a personal login along with the textbox for input and output (1&4).
Macros run	Start program and login.	Starting the program displays the login screen.	A neat and presentable GUI (6).


Flow Charted Pseudocode for Initial Login of the program
Start
macroAuto
(loads the login screen)
LoginScreen
Display "Login Fail"
Get userN from user
Get passWd from user
Does userN == COMPUSER?
&
Does passWd == COMPPASS?
No
Yes
Run macroMen
(loads Main menu)

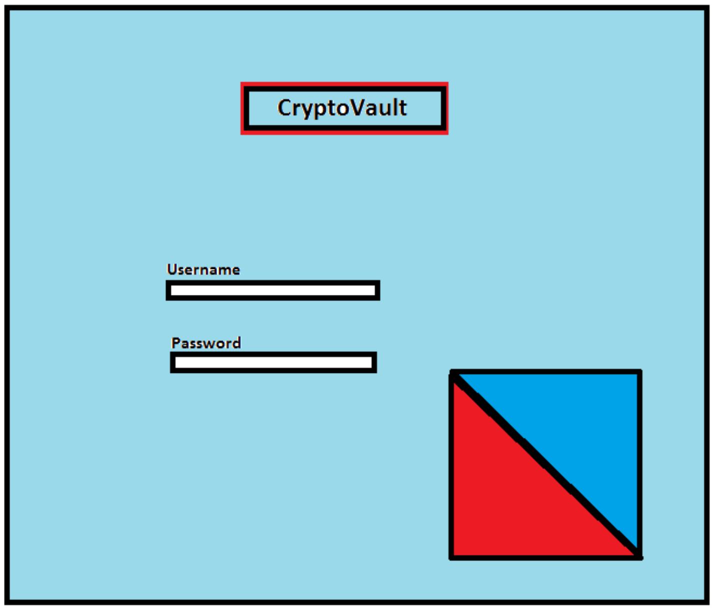


Display inStrucMorse**
Load a method key
Get inMess from user placed in textBox1
Display “Characters are not approved”
Flow Charted Pseudocode for *Morse code encrypting and decrypting
Is inMess valid?
No
Yes
Is cryptoDec == Encrypt? Or is cryptoDec == Decrypt?
Encrypt
Decrypt
Run inMess into Encrypt method
Run inMess into Decrypt method
Display outMessEncry in textBox2
Return to menu
Display outMessDecry in textbox2

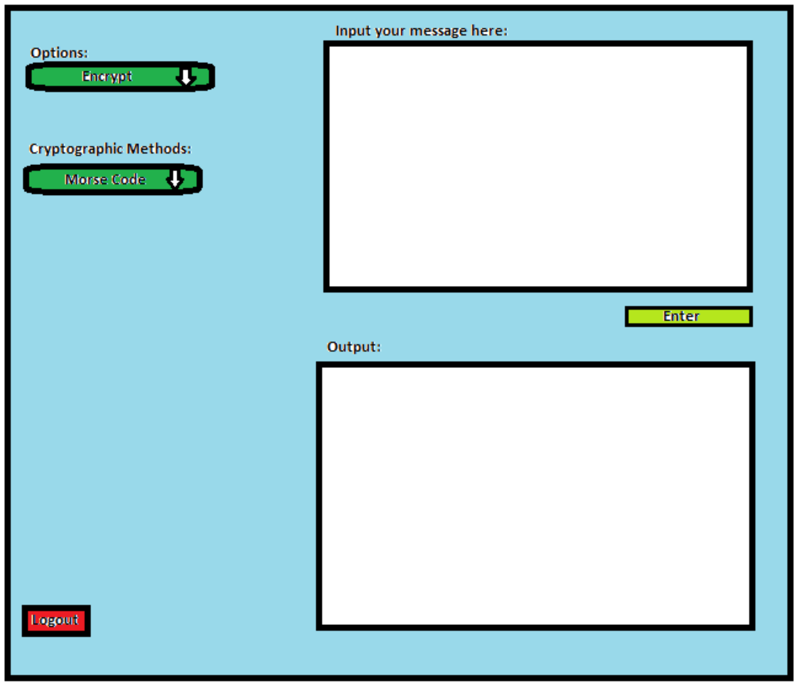
*Note: The Caesar Shift Cipher, Vigenère, and ZigZag writing methods follow this same flow chart.

** Other functions that can replace inStrucMorse: inStrucZig, inStrucCae, inStrucVig,

Example Login Screen for the Program:



Example GUI for the workspace:



Algorithm/Pseudocode:

Encrypt:

- Each method for encrypting and decrypting is located in the Cipher class.

```
If (Encrypt == morseEn)
             Read inMess;
             Find inMess characters in array charAc;
             If(found)
             int foundIndex = indexOf the inMess characters;
             //Sets foundIndex to the position of the characters found in
the charAc array.


               String enCiph = morse[foundIndex];//Finds the
corresponding Morse object and sets it to enCiph.
               Output (enCiph) in textBox2;
*Else If (Encrypt == caeSarEn)
             Read inMess;
             Find inMess characters in array caesar;
             If(found)
             Read key;
             Shift every character in the caesar array
             forward(adding)by key;
             int foundIndex = indexOf the inMess characters;
             String enCiph = morse[foundIndex];
             Output(enCiph) in textbox2; //Outputs each character one
             by one.
Else If (Encrypt = zigEn)
             Read inMess;
             If(indexOf a character is 0 or odd)
               Place into first array;
             Else If(indexOf a character is even)
```

```
                  Place into second array;
                Output (elements of first array)in textbox 2;
                Output (elements of second array) in textbox 2; //In
one line
```

Decrypt:

```
If (Encrypt == morseDe)
                Read inMess;
                Find inMess characters in array morse;
                If(found)
                int foundIndex = indexOf the inMess characters;
                //Sets foundIndex to the position of the characters found in
the morse array.

                  String deCiph = charAc[foundIndex];//Finds the
corresponding Morse object and sets it to enCiph.
                  Output (deCiph) in textBox2;
*Else If (Decrypt == caeSarDe)
                Read inMess;
                Find inMess characters in array caesar;
                If(found)
                Read key;
                Shift every character in the caesar array
                backward(subtracting) by key;
                int foundIndex = indexOf the inMess characters;
                String deCiph = morse[foundIndex];
                Output(deCiph) in textbox2; //Outputs each character one
                by one.
```

*The Vignere methods are done this way as well, however, this time each character that the user inputs will be enciphered or deciphered according to a keyword instead of a keyletter. For example, if the keyword is LEMON, the first letter of the user’s input is enciphered using ‘L’ as a key letter (meaning the cipher is shifted so that any ‘As’ in the users input will be enciphered into an ‘L’, ‘Bs’ to M, and so on), the second letter will be enciphered using ‘E’ as a key letter, and so on. This continues throughout the whole message and repeats the letters of the keyword over and over again.

Morse code/Encrypt:

```
Initialize morse[‘.-‘, ‘-…’, ‘-.-.’, ‘-..’, ‘.’, ‘..-.’, ‘--.’,
‘….’, ‘..’,

‘.---‘, ‘-.-‘, ‘.-..’, ‘- -‘, ‘-.’, ‘---‘, ‘.--.’, ‘--.-‘, ‘.-.’,
‘…’, ‘-‘, ‘..-‘, ‘…-‘, ‘.- -‘, ‘-..-‘, ‘-.- -‘, ‘--..’, ‘.----‘,
‘..---‘, ‘…--‘, ‘….-‘, ‘…..’, ‘-….’, ‘--…’, ‘---..’, ‘----.’,’--
---‘]

Initialize charAc[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’,
‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’,
‘W’, ‘X’, ‘Y’, ‘Z’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’,
‘0’] //Arrays correspond their indexes

Display inStrucMorse; //Displays instructions for Morse code

Get inMess from user in textBox1;

Encrypt(inMess);
```

Morse code/Decrypt:

```
Initialize morse[‘.-‘, ‘-…’, ‘-.-.’, ‘-..’, ‘.’, ‘..-.’, ‘--.’,
‘….’, ‘..’,

‘.---‘, ‘-.-‘, ‘.-..’, ‘- -‘, ‘-.’, ‘---‘, ‘.--.’, ‘--.-‘, ‘.-.’,
‘…’, ‘-‘, ‘..-‘, ‘…-‘, ‘.- -‘, ‘-..-‘, ‘-.- -‘, ‘--..’, ‘.----‘,
‘..---‘, ‘…--‘, ‘….-‘, ‘…..’, ‘-….’, ‘--…’, ‘---..’, ‘----.’,’--
---‘]

Initialize charAc[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’,
‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’,
‘W’, ‘X’, ‘Y’, ‘Z’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’,
‘0’] //Arrays correspond their indexes

Display inStrucMorse; //Displays instructions for Morse code

Get inMess from user in textBox1;

Decrypt(inMess);
```

Caesar/Encrypt:

```
Initialize caesar[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’,
‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’,
‘W’, ‘X’, ‘Y’, ‘Z’]

Initialize key;

Display inStrucCae;

Get inMess from user in textBox1;

Get key from user; //If the user chooses to do either a Caesar or Vignere
method, a new drop-down menu appears that lets the user choose what key they
would like.


Encrypt(inMess);
```

Caesar/Decrypt:

```
Initialize caesar[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’,
‘J’, ‘K’, ‘L’, ‘M’, ‘N’, ‘O’, ‘P’, ‘Q’, ‘R’, ‘S’, ‘T’, ‘U’, ‘V’,
‘W’, ‘X’, ‘Y’, ‘Z’]

Initialize key;

Display inStrucCae;

Get inMess from user in textBox1;

Get key from user;

Decrypt(inMess);
```

ZigZag/Encrypt:

```
Initialize first[];
Initialize second[];
Display inStrucZig
Get inMess from user in textBox1;
Encrypt(inMess);
```

ZigZag/Decrypt:

```
Initialize first[];
Initialize second[];
Display inStrucZig
Get inMess from user in textBox1;
Decrypt(inMess);
```