——Validate Cipher——

Testing by inputting own cipher:

```
*Assignment> let testTrueCipher = "QWERTYUIOPASDFGHJKLZXCVBNM"
*Assignment> validateCipher alphabet testTrueCipher
True
```

Testing by inputing own cipher with a lowercase letter

```
*Assignment> let testFalseCipherOne = "qWERTYUIOPASDFGHJKLZXCVBNM"
*Assignment> validateCipher alphabet testFalseCipherOne
False
```

Testing by inputting own cipher with all lowercase letters

```
*Assignment> let testFalseCipherTwo = "qwertyuiopasdfghjklzxcvbnm"
*Assignment> validateCipher alphabet testFalseCipherTwo
False
```

Testing by inputting own empty cipher

```
*Assignment> let testFalseCipherThree = ""
*Assignment> validateCipher alphabet testFalseCipherThree
False
```

Testing by using the cipher declared in the code (“EKMFLGDQVZNTOWYHXUSPAIBRCJ”)

```
*Assignment> validateCipher alphabet cipher
True
```

Testing what would happen if no cipher is given

```
*Assignment> validateCipher alphabet ""
False
```

— —Offset——

This shows what the original cipher looks like

```
*Assignment> cipher
“EKMFLGDQVZNTOWYHXUSPAIBRCJ"
```

Test showing the cipher offset to the right by 4

```
*Assignment> offset cipher 4
“BRCJEKMFLGDQVZNTOWYHXUSPAI"
```

Test showing the cipher offset to the right by 15

```
*Assignment> offset cipher 15
“TOWYHXUSPAIBRCJEKMFLGDQVZN"
```

——Encode——

Test showing that encode works for different capitalised letters and for different offset values.

```
*Assignment> encode 'H' cipher 5
‘M'
```

```
*Assignment> encode 'H' cipher 0
‘Q'
```

```
*Assignment> encode 'K' cipher 1
‘Z'
```

——Encode Message——

Test showing that encodeMessage works with any length string and for different offsets.

```
*Assignment> encodeMessage "HELLOWORLD" cipher 0
“QLTTYBYUTF"
```

```
*Assignment> encodeMessage "HELLOWORLD" cipher 5
“MJDDZUZODC”
```

This test shows the mapping of the alphabet to the offsetted cipher

```
*Assignment> encodeMessage "ABCDEFGHIJKLMNOPQRSTUVWXYZ" cipher 6
"AIBRCJEKMFLGDQVZNTOWYHXUSP"
```

——Reverse Encode——

Test using the results from the Encode example above to show that the function correctly maps the ciphered characters back.

```
*Assignment> reverseEncode 'M' cipher 5
‘H'
```

```
*Assignment> reverseEncode 'Q' cipher 0
‘H'
```

```
*Assignment> reverseEncode 'Z' cipher 1
'K'
```

——Reverse Encode Message——

Test using the results from Encode Message above to show that this function correctly maps the ciphered message back to the original message.

```
*Assignment> reverseEncodeMessage "QLTTYBYUTF" cipher 0
“HELLOWORLD"
```

```
*Assignment> reverseEncodeMessage "MJDDZUZODC" cipher 5
“HELLOWORLD"
```

```
*Assignment> reverseEncodeMessage "AIBRCJEKMFLGDQVZNTOWYHXUSP"
cipher 6
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

——Letter Stats——

These tests show that the function correctly determines the percentage of each letter in the message.

```
*Assignment> letterStats "AAA"
[(‘A',100)]
```

```
*Assignment> letterStats "AAAB"
[(‘A',75),('B',25)]
```

```
*Assignment> letterStats "ALPHABET"
[(‘A',25),('B',12),('E',12),('H',12),('L',12),('P',12),('T',12)]
```

——Partial Decode——

Here I tested the partialDecode on a small string to see if it worked.

```
*Assignment> partialDecode "HELLOWORLD" [('H', 'K'), ('E', 'T'),
('L', 'R'), ('O', 'C'), ('W', 'A'), ('R', 'J'), ('D', 'X')]
“ktrrcacjrx"
```

Here I tested what would happen if you provided an empty message to partialDecode. An empty string is returned.

```
*Assignment> partialDecode "" [('A','C'), ('N','L')]
""
```

I then went onto decoding the mystery text file. I started by taking the end four letters as they would correspond to STOP. I then went on to find vowels as that would make guessing the rest of the words a lot easier.

```
*Assignment> partialDecode mystery [('V','P'), ('N','L'),
('U','Z'), ('J','T'), ('B','B'), ('F','O'), ('S','T'), ('X','A'),
('Z','K'), ('D','U'), ('H','D'), ('R','Y'), ('A','S'), ('K','V'),
('T','F'), ('Y','N'), ('M','G'), ('E','R'), ('P','M'), ('C','H'),
('Q','I'), ('W','E'), ('L','C')]
"itseasytobreakasubstitutioncipherprovidedyouhavealongenoughmessag
estopletsmakethisonealittlebitlongerstopokitshouldbetherightsortof
sizenotstopmaybenotletsincreasethemessagelengthabitmorestop"
```