Chp.8 – String Encoding

The goal of this activity is to create a program that takes in a string and encodes it. Be sure to create a header comment with the title of this program, your name, and the date, along with a short summary in your own words about the purpose of your program. Your program should be made up of the following steps:

Part 1 – Ask the user for a string to encode. The expectation that if the sentence contains many words, they will be separated by a space. Once you run your program, the questions for the user should be formatted as follows:

```
Please input your sentence: 
```

Part 2 – Take the string sentence the user gave and divide it up into a list(string slicing, which is covered in chapter 8) of all of the words in the sentence. Each item in the list should be each word in the string sentence that is separated by a space.

Part 3 – Using the list of all of the words in the sentence you created, determine the total number of words in this list.

Part 4 – Now you will create a loop to check every word in the list:

Part 4a – Create a new variable and set it equal to the word in the list that you are currently checking in the loop.

Part 4b – Check if this current word is not an empty string (ie is not ‘’). As long as the word is not an empty string, you can continue checking the word:

Part 4c – Now check whether the word is made up of only one character. As long as the word contains more than one character, continue checking the word:

Part 4d – Create another new variable. This new variable should be set to the current word you’re checking AFTER all of the following encoding manipulations have been completed to it. Take the first character of the word you’re currently checking and move it instead to the very end of the word. For example, “hello” would become “elloh”. Then add “ay” to the end of the manipulated word. For example, “elloh” would become “ellohay”. So if the original word you were checking was “hello”, the new variable you created should be set to “ellohay”.

Part 4e – In the event that Part 4c is not true, that the word you are currently checking does not contain more than one character, simply add “ay” to its end. Remember to set the new variable you created in Part 4d to this newly encoded manipulated word. For example, if the word you’re checking is “h”, since it only has one character only “ay” should be added to its end. So if the original word you were checking was “h”, the new variable you created in Part 4c should be set to “hay”.

Part 4f – Now make sure to update the word in the list that you were checking with this new encoded version of the word that you created.

Part 5 – Finally, recombine each of the words in the list into one single string sentence again, with a space separating each word in the sentence. Then display the newly encoded sentence (however the _____ will be replaced with the correctly encoded sentence):

```
The encoded sentence is: __________
```

Overall note: This encoding process does not differentiate between lowercase letters, uppercase letters, and symbols, so the encoded message will still include each part of the sentence as it was originally written, just slightly jumbled up.

Example Test Case:

If the user's given sentence is "Hello world!", then your program should display the following encoded version of the sentence:

```
Please input your sentence: Hello World!
The encoded sentence is: elloHay orld!Way
```