Ryan Christopher
AP Computer Science Principles

Create Performance Task Mock Write Up:

3a-
Shakespeare Bot is a program created to translate words and sentences into a phrase that could be read as if it were from a poem or play written by Shakespeare. Its purpose is to generate amusing translations that combine current language patterns with old English terms often found in Shakespearian literature. Mainly created for entertainment, the program takes a piece of text as input from the user, and checks if there are any terms that can be changed. There is a dictionary of common Shakespearian terms in the program, and every word from the user input that also exists in the dictionary is translated to an old English version of the original term. The fully translated text is then is stored as a new sentence and displayed as output.

3b-
Code Segment 1-

```
# instructions for user
animateText(colorText("Enter the text you want translated: ","green"), 0.01)
# get user input for text to translate
text = input("")

# use a nested loop to only allow text that is at least one character in length
while len(text) < 1:
     animateText(colorText("Please enter a piece of text to translate: ", "green"), 0.01)
     text = input("")

# bring the input to all lower case
text = text.lower()

# split into a list of each word and punctuation mark
# ex: "hi guys!" ⟶ ["hi", " ", "guys" "!", ""]
text_list = re.split("([^a-zA-Z0-9'])", text)

# call the translate function and store it in a variable to display
translated_text = translate(text_list)
```

Code Segment 2-

```
# selection - checking if the string is in the
# dictionary of shakespearian words, and if it is
# then change the word to it's shakespearian value
if textArray[i] in shakespeare_words:
     textArray[i] = shakespeare_words[textArray[i]]
     translated_text += colorText(textArray[i], 'green')

# otherwise, concatenate (or add) the string on to
# the translated text variable as is with no changes
else:
     translated_text += textArray[i]
```

In this program, the list being created by the user is stored as the variable name “text_list”. This list is generated by using the function re.split from the regular expressions library. The function takes the variable “text” that is input from the user, and generates an array that separates each piece of the input and stores them as elements. In “text_list”, the elements are strings that either contain a word, space, or punctuation mark. To manage complexity, the “text_list” is then passed to a function with the parameter named “textArray”, and compared one element at a time to the dictionary called “shakespeare_words” of stored words with their matching Shakespearian term. Were it not for the array separating each word, space, and punctuation mark, translating the terms would be much more difficult as the entire string would need to be sorted through one character at a time.

3c-

Function translate(textArray) -

```
# function with one paramater taking an array, and
# iterating through the array
def translate(textArray):

     # get length of array of text
     terms = len(textArray)

     # create empty string to store new text
     translated_text = ""

     # iterate through the textArray values
     for i in range(terms):

          # selection - checking if the string is in the
          # dictionary of shakespearian words, and if it is
          # then change the word to it's shakespearian value
          if textArray[i] in shakespeare_words:
               textArray[i] = shakespeare_words[textArray[i]]
               translated_text += colorText(textArray[i], 'green')

          # otherwise, concatenate (or add) the string on to
          # the translated text variable as is with no changes
          else:
               translated_text += textArray[i]

     # at the end of the function, return the full translated text back
     # to be displayed
     return translated_text
```

Example of function being called-

```
# split into a list of each word and punctuation mark
# ex: "hi guys!" ⟶ ["hi", " ", "guys" "!", ""]
text_list = re.split("([^a-zA-Z0-9'])", text)

# call the translate function and store it in a variable to display
translated_text = translate(text_list)
```

The procedure “translate(textArray)” helps to contribute to the overall functionality of the program by taking the array containing the user’s phrase that has been separated, and iterating throughout the entire array to determine which words can be translated. By using the python method “in”, it will return True if the element of the list exists in the dictionary, and False if the string does not exist in the dictionary.

When a user enters a phrase they want translated, it is stored as an array containing words, spaces, and punctuation marks. Then, the array that is generated from the user’s input is passed to the function translate(textArray) which takes one parameter as a list. In order to iterate properly, the length of the array is stored as “terms” obtained through the python method len(). The next step in the algorithm to this function is to create an empty string called “translated_text”, which will be used to concatenate each value that is either translated or kept the same. The function then iterates through the entire “textArray” list using a for loop. The loop utilizes the range function with the terms variable, and for each element at the position of “i”, selection using conditional statements are used. The conditional statements will check if the element at the current position in “textArray” exists in the “shakespeare_words” dictionary, either returning True or False. If it is True, then the shakespearian term from the dictionary will be added to the “translated_text” variable, and if the term does not exist and the statement is False, the else statement adds the original term to the “translated_text” variable. Once the array has been iterated through, the “translated_text” string is returned to be displayed to the user by printing the variable name “translated_text”.

3d-

The test cases are two phrases, the first one being only terms that are in the dictionary, and the second being terms that exclusively do not exist in the dictionary to obtain different results. For both times the function is called, the stored array will be compared to the dictionary one element at a time, and either it will exist in the dictionary and will return True along with replacing the English term with the shakespearian term, or False in which the original term will remain. The first call, with the string (“where are you”) results in the output of (“whence art thou”), as all three words “where”, “are”, and “you” exist in the dictionary. The second call, with the string (“we like video games”) will return (“we like video games”) due to none of the terms existing in the dictionary.