//Project 2: Pseudocode Python

// Bailie Allemand, Tiago Araujo, Connor Kasarda, Jeffrey Michaelis, Matt Sharp

This program translates messages from English to Morse code, as well as from Morse code to English using the coding language Python

Node class creates nodes that will contain a character value as well as a right and left child; both children are defaulted to none in order to be used to create the binary tree necessary for the Morse to English translation of a given message.

Function insert(self, path, letter)

Function inserts a letter into the binary tree at its correct position

If the first character in path is equal to (‘.’)

If the length of the path is greater than 1

Path will equal all elements of path besides the first element

If the node of the binary tree at the given position does not have a left child

Create left child with empty character value

Recursively call Insert function with the new path value

Else (meaning the path is 1 symbol)

If the node of the binary tree at the given position does not have a left child

Create left child with empty character value

Else

If the left node is empty

Add the given letter to the node

Else

Print an error message saying that a letter already exists in that node

Else if the first character in path is equal to (‘-‘)

If the length of the path is greater than 1

Path will equal all elements of path besides the first element

If the node of the binary tree at the given position does not have a right child

Create right child with empty character value

Recursively call Insert function with the new path value

Else (meaning the path is 1 symbol)

If the node of the binary tree at the given position does not have a right child

Create right child with empty character value

Else

If the right node is empty

Add the given letter to the node

Else

Print an error message saying that a letter already exists in that node

End

Function find\_letter(self, hidden\_letter\_path)

Function finds the character in the binary tree that corresponds to the Morse code given

If the first character in hidden\_letter\_path is equal to (‘.’)

If the length of the hidden\_letter\_path is greater than 1

hidden\_letter\_path will equal all elements of hidden\_letter\_path besides the first element

Returns a recursive call of find\_letter using the new hidden\_letter\_path and the left child of the previous node in the binary tree

Else

Returns the letter at the left child node from the current node in the binary tree

Elseif the first character in hidden\_letter\_path is equal to (‘-’)

If the length of the hidden\_letter\_path is greater than 1

hidden\_letter\_path will equal all elements of hidden\_letter\_path besides the first element

Returns a recursive call of find\_letter using the new hidden\_letter\_path and the right child of the previous node in the binary tree

Else

Returns the letter at the right child node from the current node in the binary tree

End

Function English2Morse

Function translates a file from English to Morse code, inputting the translation into a new file

Create an empty dictionary, 2 empty strings, and an empty array

Create a Boolean exception2 value defaulted to True

If the user inserted new Morse symbols

Open the new Morse table

Else

Open the default Morse table

For all lines in the Morse table file

Add the values in the line to the dictionary, with the first character being the key, and all characters after the spaces until the end of the line being the value for that given dictionary entry

Close the Morse table being used

While a valid file name needs to be entered

Try

Prompt the user for a file name to read in for translation

Open the file to be translated

Break from the loop

Except

Prompt the user by telling them the file does not exist and they can try again inputting a valid file

Creates a new file that will contain the message after it is translated from English to Morse

Invariant: line is always less than or equal to the number of lines in the English to Morse file

For all lines in the file being translated from

While the first character in a line equals a space (‘ ‘)

Write a space into the new file

Make the line equal all values of the line besides the first value

Split the line into individual words and insert into the array

If the array of words is not empty

If file read in already contains Morse values

Sets an exception2 value to false

Assigns an error message

Has shell print out an error message telling the user that the file contains no letters

Break from the loop

Else

# Invariant: word is always less than or equal to number of words in words list

For each word in the word list array

# Invariant: letter is always less than or equal to the number of letters in word

For each letter in each word

Try

If the letter is not uppercase

Based off the dictionary, using the letter as the key, add the values dots and dashes to a string, followed by a space

Else

Make the letter lowercase

Starting with a (‘\*’) symbol to detonate that the letter is uppercase, based off the dictionary, using the letter as the key, add the values dots and dashes to a string, followed by a space

Except (letter not found)

Boolean value created at the beginning is set to false

Add character to string of characters followed by (‘,’), the list being the characters that are not found in the Morse table

Reasoning for not translating letter is given (letter not found)

Write the translated word to the new file

Make the string that contained the translated word an empty string again.

Write 3 consecutive spaces into the new file to separate words (3 spaces instead of 1, because 1 space separated letters)

Make exceptions set to true

Write a new line to the translated file

Else

Write a new line to the translated file

Close the file that reads in the message to be translated

Close the file that was created to contain the translated message

If exception is true

If exception2 is true

Shell prints out a success message

Else

Shell prints out a partial success message, saying some symbols could not be found in the Morse table

Else

Shell prints out a failure message along with the reason why.

End

Function Morse2English

Function translates a file from Morse code to English, inputting the translation into a new file

Create an empty root node.

Create a count, defaulting to 1

Create a Boolean exception2 value defaulted to True

If the user inserted new Morse symbols

Open the new Morse table

Else

Open the default Morse table

For all lines in the Morse table file

Call the insert function. This will input the values from the Morse table into the correct position in a binary tree.

Close the Morse table being used

While a valid file name needs to be entered

Try

Prompt the user for a file name to read in for translation

Open the file to be translated

Break from the loop

Except

Prompt the user by telling them the file does not exist and they can try again inputting a valid file

Create a new file that will contain the message after it is translated from Morse to English

#Invariant: line is always less than or equal to the number of lines in hidden\_morse\_file

For all lines in the file being translated from

While the first character in a line equals a space (‘ ‘)

Write a space into the new file

Make the line equal all values of the line besides the first value

Split the line into individual letters (each set of letters is separated by one space)

Correct the last letter in the file

If array of Morse characters is not empty

If file contains Morse Values

# Invariant: hidden\_letter\_path is always less than or equal to the number of dashes and dots in morse\_letter

For each set of Morse Characters in the array of Morse Characters

Try

If the set of Morse characters is not empty

If the first character is not a (‘\*’)

Search the binary tree for the equivalent letter and write the letter into the new file

Else

Make a new string, excluding the first character

Search the binary tree for the equivalent letter and write it capitalized in the new file

Else

Increment the count up by 1

If the count is divisible by 2 with no remainders

Write to spaces to the file to indicate the end of the word

Else

Do nothing

Except

Shell prints out error saying Morse code was not found in the file

The Boolean expcetion2 is set to false

A reason is given for why the translation failed

Write a new line to the translated file

Set an exception value to be true

Else

Set an exception value to be false

Shell prints out error saying the file entered is not a Morse code file

A reason is given to why the translation failed

Break from the loop

Else

Write a new line to the translated file

Close the file that reads in the message to be translated

Close the file that was created to contain the translated message

If exception is true

If exception2 is true

Shell prints out success message

Else

Shell prints out a partial success message, saying some of the Morse code was ignored due to being invalid

Else

Shell prints out a failure message along with the reason why

End

Function newMorseTable

Function allows the user to change the symbols of Morse code table from dots and dashes to others

Declare 3 variables to be used

Set the first variable to be equal to (‘\*’)

While the first variable is equal to (‘\*’) or the length of the variable is greater than 1

First variable equals new variable inputted by user

If the first character of the variable is (‘\*’)

Shell tells user that the symbol must be different

Elseif the length of the variable is greater than 1

Shell tells user that the symbol can only be one character

Second variable is set to the value of the first variable

While the first and second variable are equal or the second variable is equal to (‘\*’) or the length of the second variable is greater than 1

Second variable equals new variable inputted by user

If the first variable equals the second variable

Shell tells user that the symbols must be different

ElseIf the first character of the variable is (‘\*’)

Shell tells user that the symbol must be different

Elseif the length of the variable is greater than 1

Shell tells user that the symbol can only be one character

Else

Do nothing

Opens the original Morse code table file

Creates new file for the new Morse code table

# Invariant: line is always less than or equal to the number of lines in the Morse\_Table file

For all lines in the Morse table file

Write the old Morse code into the new file (excluding the dots and dashes)

# Invariant: i is always a dash or a dot

For all elements in the line excluding the starting letter

If the element is a (‘.’)

Replace with the new first variable in the new file

Elseif the element is a (‘-‘)

Replace with the new second variable in the new file

Else

Do nothing

Write a new line in the new file

Close the original Morse code table file

Close the new Morse code table file

Set the newM to true, meaning you did change the symbols

Call the Enlgihs2Morse function

End

In the Main Function

Declare two variables. One takes the value (‘.’) while the other takes the value (‘-‘)

Declare a variable that holds the truth value of whether or not the user has changed the values for the Morse symbols. This variable should be defaulted to false.

Print a welcome message to the user (Hello, Welcome to the Morse Code Project)

##Add Invariant while \_\_\_\_\_\_

Prompt the user to choose what they would like to do with the program (Translate Morse to English, Translate English to Morse, change the given Morse code symbols, or exit the program)

If the user choses to translate a message from Morse to English

Call the function Morse2English

ElseIf the user choses to translate a message from English to Morse

Call the function English2Morse

Elseif the user choses to exit the program

Print a goodbye message

Break from the loop

Elseif the user choses to change the Morse code symbol(s)

Call the function newMorseTable

Else

Output an error message

End