//Project 2: PsuedoCode C++

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

This program translates messages from English to Morse code, as well as from Morse code to English.

Structure Node creates a node that contains a character value and has a left and right child node; both defaulted to NULL in order to be used to create the binary tree used for the Morse to English translation.

Function addLetter(character variable, string variable, Node variable)

Function adds letters and characters to the binary tree that is used for decoding from Morse code to English.

If string variable size is equal to 0

Then place char variable as the value for the current Node

Else if a (‘.’) is encounter at the current position in the string variable

Moves to the current Nodes left child (default starts at the root of the tree)

If the current node does not have a left child

Create a left child for the node

Recursively calls function by removing first character from string variable to keep running until the end of the string variable size.

Else if a (‘-’) is encounter at the current position in the string variable

Moves to the current Nodes right child (default starts at root of tree)

If the current node does not have a right child

Create a right child for the node

Recursively calls function by removing first character from string variable to keep running until the end of the string variable size.

End

Function decodeTreeConstructor(string variable, Node variable)

Function constructs the Binary tree used for the Morse to English translation in this program.

` Opens file used to construct tree.

If the file is open

While there is still a line to read in the file

The character to be inserted into a specific node will be assigned to a char variable

The string of dots and dashes will be assigned to a string variable

Will call the addLetter Function in order to insert the character in correct position in the binary tree. Function will hold the values of the character, string, and the root node as its variables.

Closes the file used to construct the tree.

End

Function findLetter(string variable, Node Variable)

Function finds a specific character represented by the Morse code string that is given

If the size of the Morse code string is 0

The character at the current Node is the letter represented by the Morse code string.

Prints out the letter found.

Else if a (‘.’) is encountered at the current position in the string variable

If a left child does not exist from the current node

Prints “Error” message to tell user that a letter was not found for that Morse code string

Else Recursively calls function by removing first character from string variable to keep running until the end of the string variable size, and by moving to the right child of the current node.

Else if a (‘-’) is encountered at the current position in the string variable

If a right child does not exist from the current node

Print “Error” message to tell user that a letter was not found for that Morse code string

Else Recursively calls function by removing first character from string variable to keep running until the end of the string variable size, and by moving to the right child of the current node.

End

Function E2M(string variable1, string variable2)

Function translates a message from English into Morse code.

Create a string variable that will hold values of the dots and dashes that are given to a specific character

Open a file whose name is the value of string variable2, which should contain the message that is expected to be translated

Find the length of the message in the file and stores this value to the variable to be used later.

Create a buffer for the English message text

Read in the complete message in the file using the file length to ensure full message is read in

Open a file whose name is the value of string variable1, which should contain a table that represents the characters and how they translate into Morse code.

Find the length of the file that contains the table and stores this value to a variable for later use

Create a buffer for the Morse table text

Read in the complete Morse code table in the file using the file length to ensure that the full table is read

## Add Invariant Starting at the beginning of the message length and going until the end, incrementing by 1 each iteration

## Add Invariant Starting at the beginning of the Morse code table and going until the c current character in the message is found in the table, incrementing by 1 each iteration

If the current character from the translation file is found in the Morse table file

Make an int variable equal to 2, to skip the character that is given in the Morse code Table

#Add invariant While the Morse code tables buffer is either a (‘.’) or a (‘-‘)

If the current character in the table is a (‘.’)

Add a (‘.’) to the string holding the current characters dots and dashes

Else if the current character in the table is a (‘-‘)

Add a (‘-‘) to the string holding the current characters dots and dashes

Increment the int variable by 1 so the while loop can read in the next dot or dash (as long as there is one)

Print out the string holding the dots and dashes for the specific letter

Reset the string holding the dots and dashes back to an empty string

Break from the if

Else if there is a space next in the translation file

Print 3 spaces for clarity purposes

Reset the string holding the dots and dashes back to an empty string

Break from the ElseIf

Reset the string holding the dots and dashes back to an empty string.

Close the file holding the message to be translated.

Close the file holding the Morse Code Table

End

Function M2E(Node variable, string variable)

Function translates a message from Morse Code into English using a binary tree.

Create a string variable that will hold characters for the message being translated

Open a file whose name is the value of string variable, which should contain the message that is expected to be translated

Find the length of the message in the file and stores this value to the variable to be used later.

Create a buffer for the Morse code message text

Read in the complete message in the file using the file length to ensure full message is read in

#Add Invariant Starting at the beginning of the message length and going until the end, incrementing by 1 each iteration

If the current character is a (‘.’)

Add a (‘.’) to the string holding the dots and dashes for the current character to be decoded

ElseIf the current character is a (‘-’)

Add a (‘-’) to the string holding the dots and dashes for the current character to be decoded

Else if the next 3 characters are empty spaces, indicating the start of a new word

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Print out a space

Reset the string holding the dots and dashes back to an empty string.

Else if the current character is a space

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Reset the string holding the dots and dashes back to an empty string.

Call the findLetter function, giving to it the string holding the dots and dashes as its string variable and the root of the binary tree as its node variable.

Reset the string holding the dots and dashes back to an empty string.

Close the file holding the message to be translated

End

In the Main Function

Make a new Node that will be the beginning of the binary tree for decoding

Call decodeTreeConstructor with a string variable that is a file name that holds a Morse code table and the node variable that is the node created in main

Call E2M with the first string variable being a file name that holds a Morse code table and the second string variable being a file name that holds the message to be translated

Print a line

Call M2E with the node variable being the node created in main and the string variable being a file name that holds the message to be translated

Can call M2E and E2M as many times as necessary to carry out tests, should just follow structure above in order for clarity.

End