AUTHORS:

Jason Kwak	RUID: 126007397	netID: jgk68
Anne Whitman	RUID: 042007629	netID: alh220

NAME:

pointersorter.c, pointersorter

SYNOPSIS:

./pointersorter "<STRING>"

DESCRIPTION:

pointersorter takes a series of component strings inside one set of quotation marks separated by non-alphabetic characters and outputs them in ASCII alphabetical order (where A-Z come before a-z alphabetically), separated by a new line.

If no string is given or more than one set of quotation marks appear, an error is printed. [There should be one (and only one) input.]

If multiple delimiters are given, the program assumes they should be one delimiter and treats them as such.

See below for examples of acceptable (and unacceptable) input and output.

FUNCTIONS:

node * makeLinkedList(char * inputString, int sizeOfInput)
//This function takes in the user's input string and the size of that string and then separates it by non-alphabetic characters into separate words and puts each word into an unsorted linked list. It then returns the head of the linked list.

char ** linkedListToArray(node *head, int length)
//This function takes in the head of a linked list (created in previous function) and that list's length and creates an array. It then returns the new array.

char ** iSortArrays(char ** charArray, int length)
//This function takes in an array (created in previous function) and that array's length and sorts it using the strcmp function inside of a double for-loop. It then returns the sorted array.

void printArray(char ** wordArr, int listLength)
//This function takes in a sorted array (created in previous function) and traverses that sorted array, printing each word. Since this function is meant to provide output to the user, it returns nothing.

BACKSLASH CHARACTER & ESCAPING

If using a character that must be escaped as a delimiter from the command line (for example, "$" or "!"), you must escape that character with a backslash "\" on the command line.

EXIT STATUS

Normally, the exit status is 0 if input is valid and no errors occur during runtime, otherwise exit status will be a 1. A 1 will be returned with invalid input or when memory fails to allocate during run time.

EXAMPLE INPUT/OUTPUT:

./pointersorter “I-like&Hot dogs”
Hot
I
dogs
like

/.pointersorter “z@a%Z)A9”
A
Z
a
z

./pointersorter “anY answers added As aGreeable are alright.”
As
aGreeable
added
alright
anY
answers
are

./pointersorter “”

EXAMPLE ERRORS:

./pointersorter “I-like” “Hot dogs”
Incorrect number of parameters.
Only enter one parameter after executing the program
./pointersorter
Incorrect number of parameters
Must enter a string to split and sort after executing the program
./pointersorter “My^name8is@pointersorter”
Memory could not be allocated, please try again.
In this case there was no issue with userinput, but the operating system declined to allocate memory to run. Try your command again.

ALGORITHM

Set UserInput equal to user's input
Set inputLength equal to length of user's input
Set loop counter variable equal to 0
Set word length counter equal to 0
Set total length counter equal to 0
Set node counter equal to 0
While loop counter is less than inputLength
    If alphabetic character
        Increment word length counter
    Else
        Create and malloc new word
        Put that word into a linked list
        Increment node counter
        Set Total Length Counter += wordlength+1
Set loop counter equal to 0
Create a double array
While loop counter is less than node counter
    //Starting with head node
    Array[loop counter] = word in node
    Increment loop counter
    Move to next node
Set loop counter to 0
While loop counter is less than length of array
    While (loop counter + 1) is less than length of array
        Compare alphabetical order of two words
    Swap into alphabetical order if necessary
Set loop counter to 0
While loop counter is less than length of array
    Print word
Free all mallocs
Return