Function unit testing outputs

Extra credit attempted. see valgrind output at bottom of pdf.

Unit Testing for extract function & showList function
Output for extract and showList is as expected.
Extract function extracts valid english words out of a line and show list function displays the list

./a.out testfile1 sort1.log sort2.log

./a.out testfile1 sort1.log solution		
	l I count I	
la	13 1	
l lbasically	 1	
Icalling	I1 Î	
leach	I1 I	
Ifile 	13 I	
lfor 	11	
Ifrom	12 1	
Ifscanf	I1 I	
Ifunction	 1	
li	 1	
lin 	. 11 . İ	
lmy 	- I1 İ	
Iscanning	I1 Î	
Istring	1	
Istrings	II İ	
Ithe		
Ithen	I1 I	
lusing	I1 Î	
1		

./a.out testfile1 sort1.log sort2.log

l word	l count
la	12
llbasically	1 1
Icalling	
lcase	l 1
Idifferent	1
leach	l 1
Ifile 	13
lfor	l 1
Ifrom	12
Ifscanf	l 1
Ifunction	l 1
li !	1
lin 	l 1
lmy 	l 1
Iscanning	l 1
Istring	l 1
Istrings	l 1
Ithe	l 1
Ithen	1
lusing	l 1

./a.out testfile1 sort1.log sort2.log

į	word	I count	
la		l 11	
Ibasica	ally	l 1	
lcalling	1	1.1	i
ı	,		ľ
1			

 3
1
12
1
1
1
1
1
1
1
l 1 i
l 1 i
1
1

Unit Test for isLetter() function. Output is as expected

./a.out testfile1 sort1.log sort2.log output:

1 <- tested whether 'A' is a letter. Came back true

1 <- tested whether 'b' is a letter. Came back true

0 <- tested whether ',' is a letter. Came back false

Unit Test for addWord() function:

./a.out testfile1 sort1.log sort2.log

The output is as expected, for the first test I passed in a null head and a single word "word" a new head was created with the word in it. As for the second one, the head was not NULL and I passed in a different word and checked the count on the third pass.

output: word different

unit test for hasRepeat

Output is as expected. First test I pass in the same word that is in the list and it returns with that position. Second pass I pass a different word and it returns null. Third pass i try a repeat again and it returns with the position of the repeat just like the first pass

output:

repeat at node containing word temp is null, no repeat in list repeat at node containing word

unit test for addPos() function

The output is as expected. The function returns the pointer to the node before which the new node should be added. In the calling function, if the return pointer is checked and if it is pointing to the head node, the addFirst() method is called.

output:

passed in word b is less than current position containing a. insert after current pos.

passed in word a is greater than current position containing b. current position points to head node, add as new head node

passed in word b is greater than current position containing a. insert after current pos.

AddFirst() function unit testing

output is as expected. In the first case, the passed in head value is null and it returns the pointer to the second parameter as the start of the list. in the second case, the head is not null, and it adds the new node to the beginning of the list and returns a new head node pointer. The third test shows the same as the second.

output:

old head value is null new head value is word

old head value is word new head value is bob

old head value is bob new head value is adam

CreateNode function unit testing

output is as expected. the function is very simple. 1st test gives a word and a value of 1. 2nd test gives a word and a value of 2. 3rd test gives a word and a value of 50.

output:

this 1

is 2

the writelist() function outputs the list pointed to by the head pointer parameter into the file pointed to by the file pointer parameter

The output is as expected. It formats the list in a nice format and writes it to a file using fprintf. At the end of the function the file is closed

output:

Here is the content of the file that was written.

the filename was called sort1.log

ine filen 	ame י	was call	
 		I count	į
 a 		13	
 basicall 	у	l 1	
 calling 		l 1	
each		l 1	
 file 		13	
lfor		1	ĺ
from		12	i
fscanf		1	
function	1	11	
 i 		11	
in		l 1	
my 			
scannin			
		1	
 strings 		1	- 1
lthe		1	i
 then 		l 1	
 using 		l 1	

	stfile1 sort1	.log sort2.log
•	ord I coun	•

la	12
l lbasically	l 1
Icalling	1
lcase	- I1 Î
l ldifferent	 1
leach	I1 I
Ifile	13 İ
lfor	1
lfrom	12 1
IIfscanf	 1 1
Ifunction	1
 i	11 İ
	=
lin	1
-	1
Iscanning	1
Istring	1
Istrings	11 İ
-	
Ithe	1
Ithen	1
lusing	11
•	

./a.out testfile1 sort1.log sort2.log

l v	vord	I count
la		l 11
1		
Ibasicall	y	l 1
lcalling		14
icalling		1 1
leach		1.1
L		

lfile	13	
lfor	l 1	
Ifrom	12	
•		
Ifscanf	l 1	
Ifunction	l 1	
•		
li	l 1	
lin	l 1	
=		
lmv		
lmy	l 1	
Iscanning	 1	
IIscanning	1	
Iscanning I Istring	1 1 1	
IIscanning IIstring IIstring	1	
IIscanning IIstring IIstrings	1 1	
IIscanning IIstring IIstrings IIstrings	1 1 1	
IIscanning IIstring IIstrings IIthe	1 1 1	
IIscanning IIstring IIstrings IIthe Ithe	1 1 1	
IIscanning IIstring IIstrings IIthe IIthen	1 1 1 1	
IIscanning IIstring IIstrings IIthe IIthen I	1 1 1 1	
I Iscanning I Istring I Istrings I Ithe I Ithen I Iusing	1 1 1 1	

nodeCopy() unit test

the output of the function is as expected. the function is fairly simple. it clones a node as well as all its member variables.

output:

node to be cloned contains:

word: clone count: 2

copied node contains:

word: clone count: 2

node to be cloned contains:

word: this count 1

copied node contains:

word: this count: 1

node to be cloned contains:

word: awesome

count: 500

copied node contains:

word: awesome count: 500

.....

sortedCount function unit test

Output is as expected. the sortedCount function duplicates the list and then preforms a selection sort on the duplicated list.

output: Please excuse the formatting. Apple's text editor doesn't copy the format over very well.

l word	l count l
la	12
 from	l 2
Icalling	1
leach	1
lfile	1
lfor	1
Ibasically	 1
Ifunction	
•	I1 İ
•	11 İ
lmy	 1
Iscanning	 1
Istring	
Istrings	 1
Ithe	 1
Ithen	 1
iai.a.	 1

l word I count	
 a 14 	1
IIfile I 2	-
Ifrom I 2 	
leach I 1	- 1
basically 1 for 1 	
lcalling 1 	
Ifunction 1	- 1
lin l 1	1
lmy l 1 	
IIscanning 1 	
lstring 1 	
IIstrings 1 	اا ا
Ithe I 1	- 1
Ithen I 1	1
lusing 1 	
l word I count	
I Ifile I 4	
 a 2 	
lfrom I 2	l
leach I 1	
llbasically 1	اا ا ا
 for 1	
 calling 1 	

Ifunction	l 1	1
li	11	1
lin	l 1	
lmy	l 1	1
Iscanning	l 1	- 1
Istring	l 1	1
Istrings	11	1
Ithe	11	1
Ithen	11	1
lusing	l 1	1

Total execution time cost for program: 60.334515 seconds

valgrind output:

```
Program Execution from start to finish took: 0.088056
==16992==
==16992== HEAP SUMMARY:
==16992== in use at exit: 0 bytes in 0 blocks
==16992== total heap usage: 71 allocs, 71 frees, 3,336 bytes allocated
==16992==
==16992== All heap blocks were freed -- no leaks are possible
==16992==
==16992== For counts of detected and suppressed errors, rerun with: -v
==16992== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 2 from 2)
```