# Function unit testing outputs

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 testfile		
•	I count	i
la 	13	
Ibasically	11	1
I Icalling I leach	l 1	   
leach	l 1	   
  file 		
lfor 	l 1	-I   
Ifrom I		-I   
Ifscanf	l 1	1
•	11	Ī
-	11	-1   
lin 		-I   
lmy 		
Iscanning I		
Istring I	l 1	-1    -
Istrings 	l 1	1
Ithe 		
Ithen 		
lusing 	1	-I 
		-1

./a.ou	ut testfile	1 sort1.ld	og sort2.lo	g
1	word	I count	I	

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		

Ifile	13
l Ifor	   1
Ifrom	l 2 
lfscanf	l 1 
Ifunction	1   1 
li 	I 1
lin	11
lmv	1   1 
Iscanning	[1]
Istring	11
Istrings	1   1
Ithe	1   1 
Ithen	1   1 
lusing	   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 1
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

cool 50

\_\_\_\_\_

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

the filename was called so			
Ī	word	I count	İ
la		13	i
Ibasica		l 1	i
Icalling		l 1	i
leach		l 1	İ
Ifile		13	<sub> </sub>   
lfor		1	,   
lfrom		l 2	ĺ
Ifscanf		l 1	ĺ
Ifunctio	on	1	Ĺ
li		l 1	İ
lin		11	i
Imy		l 1	İ
İscann	ing	1	i
Istring		1	i
İstrinas	S	1	i
Ithe		1	ĺ
Ithen		l 1	i
lusing I		l 1 	i

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

			1
1	word	I count	- 1
la		12	I

lbasically	l 1	
lcalling	l 1	
lcase	1	
I Idifferent I	l 1	
leach	l 1	
  file 	13	
Ifor	l 1	
Ifrom	12	
Ifscanf	1	
Ifunction	1	
  i 	l 1	
lin 	l 1	
lmy	l 1	
Iscanning	l 1	
Istring	l 1	
Istrings	1	
Ithe	l 1	
I Ithen I	l 1	
lusing	l 1	

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

1			-
I		I count	
			-
la		l 11	
			-
Ibasica		l 1	
			-
Icalling	l	l 1	
			-
leach		l 1	
			-
Ifile		۱3	
			-

Ifor	l 1 
Ifrom	12
Ifscanf	l 1
Ifunction	   1
li	   1
lin	   1
lmy	   1
IIscanning	   1
I Istring	
	   1
	   1   1
·=	1   
lusing I	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.

	word	I count	I
la 		l 2	   
Ifrom		l 2	   
Icallir 	ng	1	i
leach		l 1	   l
  file 		l 1	   
lfor 		l 1	   l
Ibasio 		l 1	   
Ifunct	ion	1	   
  i 	1	1	   
lin 		l 1	 
lmy 		1	   
Iscan 	ning	l 1	   
Istrino	g	l 1	   
Istrino	gs	1	   
Ithe 		l 1	,   
Ithen		l 1	   
lusino 		l 1	   l
   	word		

l 14	1
12	I
12	1
l 1	- 1
l 1	- 1
l 1	
11	1
l 1	1
l 1	1
l 1	ı
l 1	1
11	
l 1	1
l 1	I
11	1
11	1
11	1
l I cou	
l 4	   
12	1
12	1
l 1	- 1
l 1	1
l 1	1
l 1 	•
	12

li	l 1	1
•		
lin	l 1	
lmy	11	- 1
Iscanning	l 1 	 l
Istring	l 1	
	l 1	1
•		
	l 1	I
Ithen	l 1	I
lusing	11	1

Total execution time cost for program: 60.334515 seconds