Design
In design, I attempted to keep things as modular as possible, whilst optimising for memory management. This is why I decided to create the Word and WordList structures without putting pointers to the function in the structures themselves, as this would have meant that for every word and every list, there would have been redundant memory usage.

For both structs created, I decided that I would use the getter functions in the code whenever there was a possibility that the struct had not been initialised, as the getter functions check that the structure is not NULL before continuing. This prevented the need to repeat checking code for NULL values in a lot of cases.

I decided to use a linked list structure for my WordList as I really did not want to have to add a word limit to my collection (except that afforded by memory). Also, I only really needed the head and tail variables, and every time a word is added the whole list needed to be checked anyway, as I needed the list in order of first appearance and so couldn't sort it without wasting a lot of memory.

I decided to go with dynamic memory management, as this would allow me to store many words on the heap and clean-up sooner than otherwise would have. An example of this is when you add a word to a list WordList_addWord checks to see if the word is already in the list, and cleans up the duplicate parameter word if it does, and returns the correct pointer.

For error management, I wanted to keep this design testable, using a form of unit-testing for the WordList and Word structures and functions. Therefore, I made use of the errno variable in certain critical points where the program would need to exit with a failure code, but I wanted to allow for any extra cleanup to be done on existing variables, and so I would set errno, and return to the caller function, as Word and WordList are supposed to be the supporting plug-in data structures, and calling exit in them would potentially stop a larger program unnecessarily.

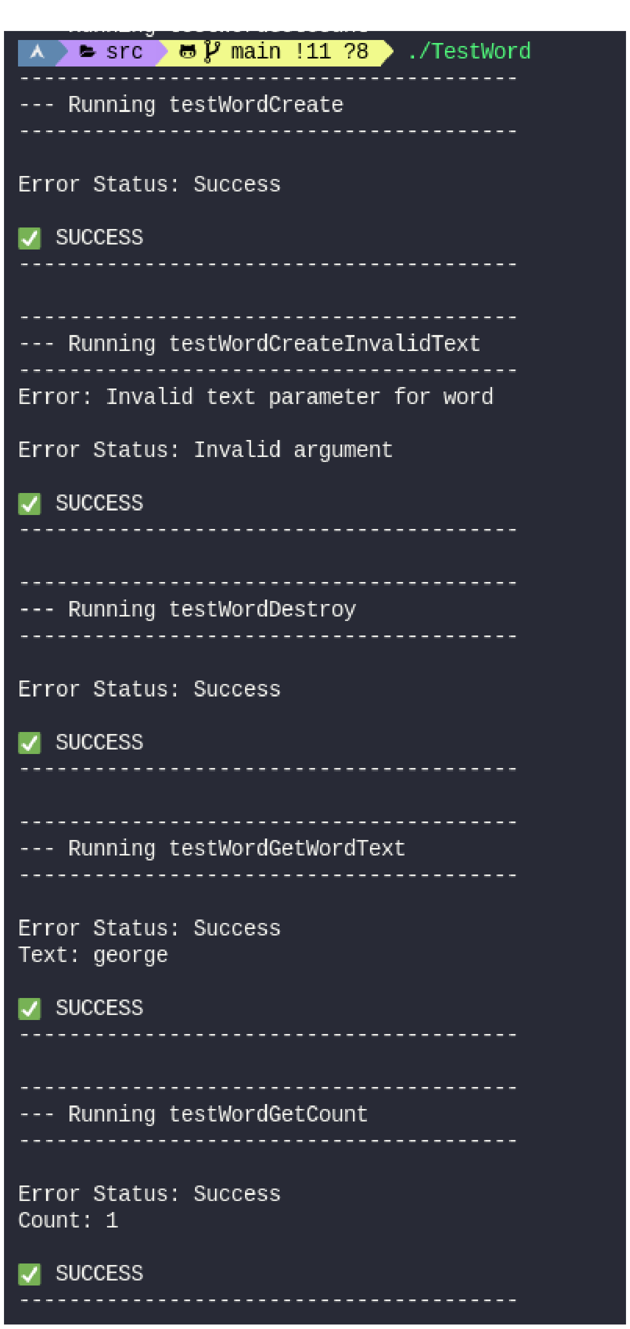
Difficulties
One of the major difficulties I had in the implementation of my code was in the clean-up stage of adding a new word into a WordList. I wanted to be able to add a Word object instead of just text. This makes it possible for further updates where two wordLists can be merged. However, the problem arose when I tested adding the same pointer to a Word twice. This issue arises from comparing the same pointer to itself. A fix I would have implemented given more time is to use memcmp, but that would have required moving the Word struct code into the word.h file as the size of the structure was unavailable in the wordList.c file. However, the application created for this spec will never attempt to enter the same pointer twice.

Testing

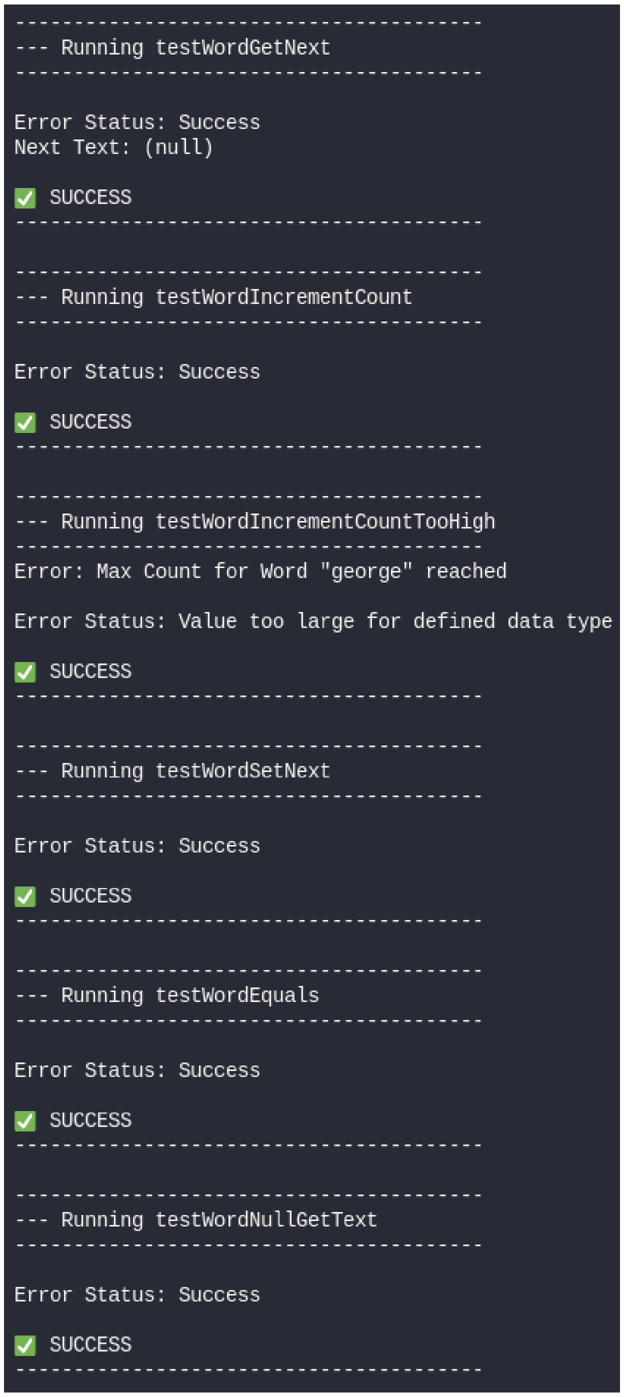
As described, I unit-tested both the Word and WordList structure and functions to ensure that they were portable and scalable. I added tests for getting their values, and setting and manipulating where necessary, ensuring that they would return the correct values, and set errno appropriately where necessary.

I also included a testing #defines and #ifdefs which lowers the limit for the maximum word count, as usually I wanted it to be the maximum integer value INT_MAX from <limits.h> but testing this would waste time. Also, didn't want to stress test my hardware by creating many words and word lists to check that the malloc error detection worked, and so I added in #define to simulate a NULL return from malloc and a setting of errno. Therefore, I could test these parts. I added in a sed command in my Makefile which would toggle on and off these #defines depending on what you are making.

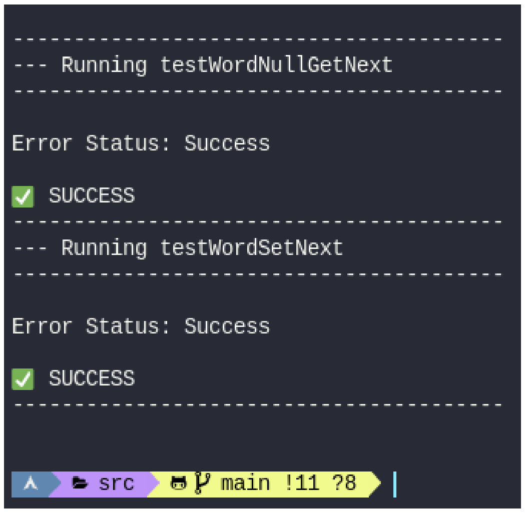
Here is the output of all of my independent testing for the Word struct and functions



```
 src  main !11 ?8  ./TestWord
----------------------------------------
--- Running testWordCreate
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordCreateInvalidText
----------------------------------------
Error: Invalid text parameter for word

Error Status: Invalid argument

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordDestroy
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordGetWordText
----------------------------------------

Error Status: Success
Text: george

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordGetCount
----------------------------------------

Error Status: Success
Count: 1

✓ SUCCESS
----------------------------------------
```

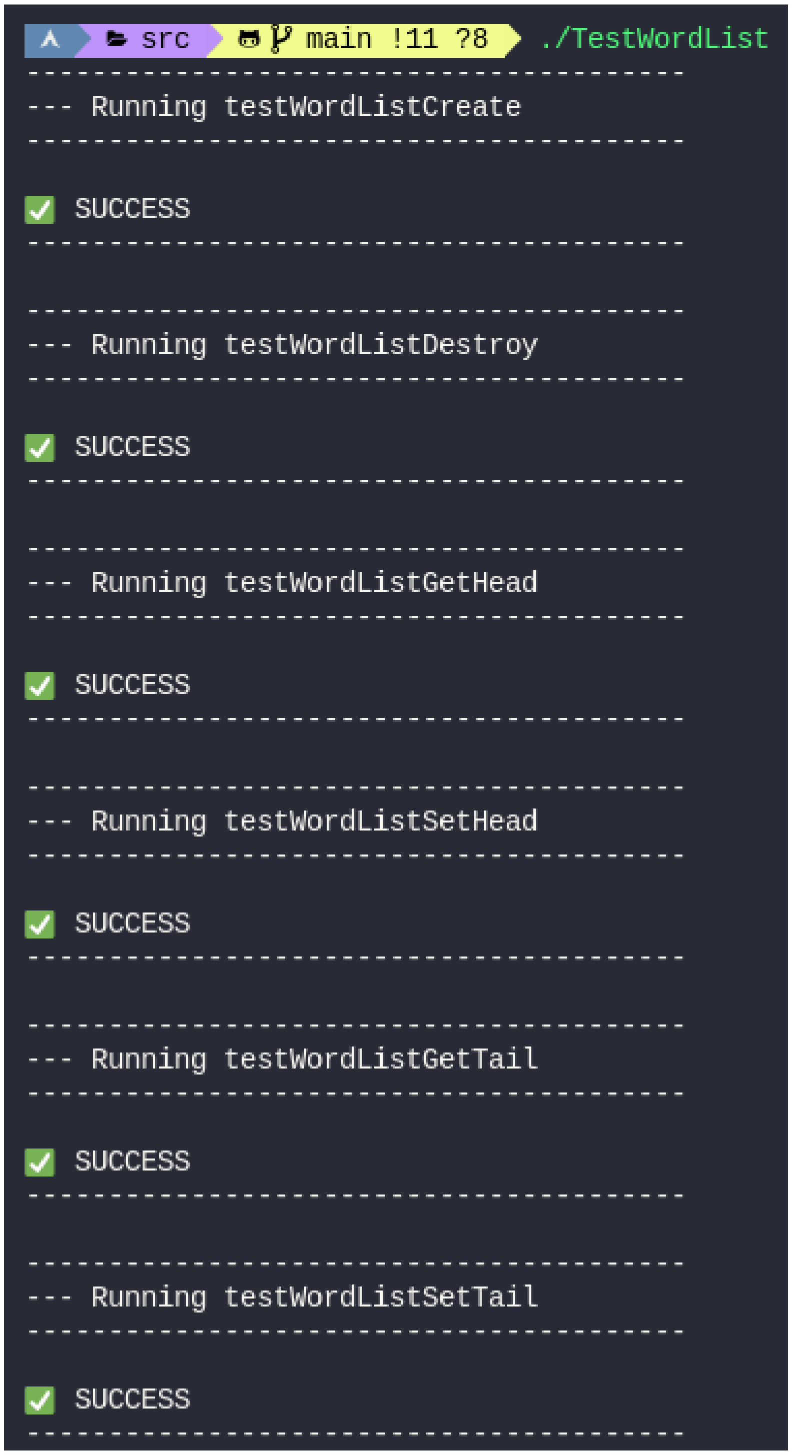


```
----------------------------------------
--- Running testWordGetNext
----------------------------------------

Error Status: Success
Next Text: (null)

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordIncrementCount
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordIncrementCountTooHigh
----------------------------------------
Error: Max Count for Word "george" reached

Error Status: Value too large for defined data type

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordSetNext
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordEquals
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordNullGetText
----------------------------------------

Error Status: Success

✓ SUCCESS
----------------------------------------
```

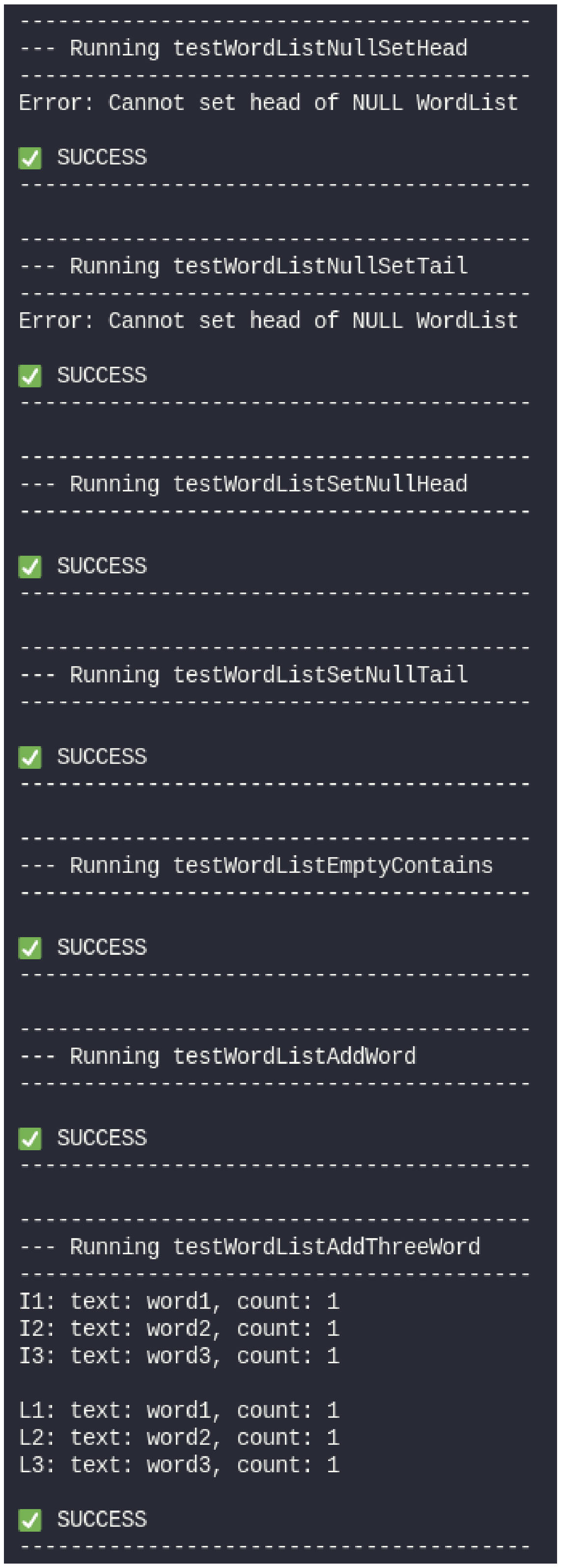


```
----------------------------------------
--- Running testWordNullGetNext
----------------------------------------

Error Status: Success

✅ SUCCESS
----------------------------------------
--- Running testWordSetNext
----------------------------------------

Error Status: Success

✅ SUCCESS
----------------------------------------


 src   main !11 ?8
```

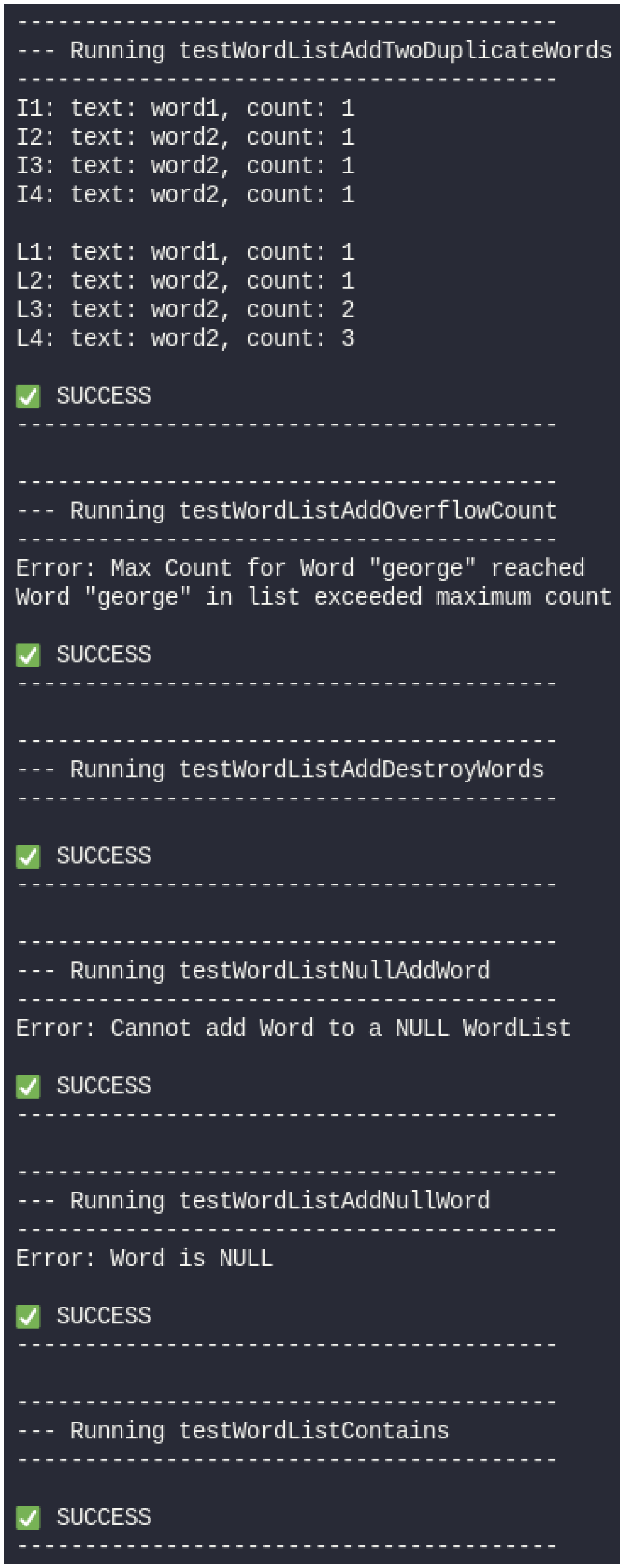
Here's my independent testing for the WordList structure



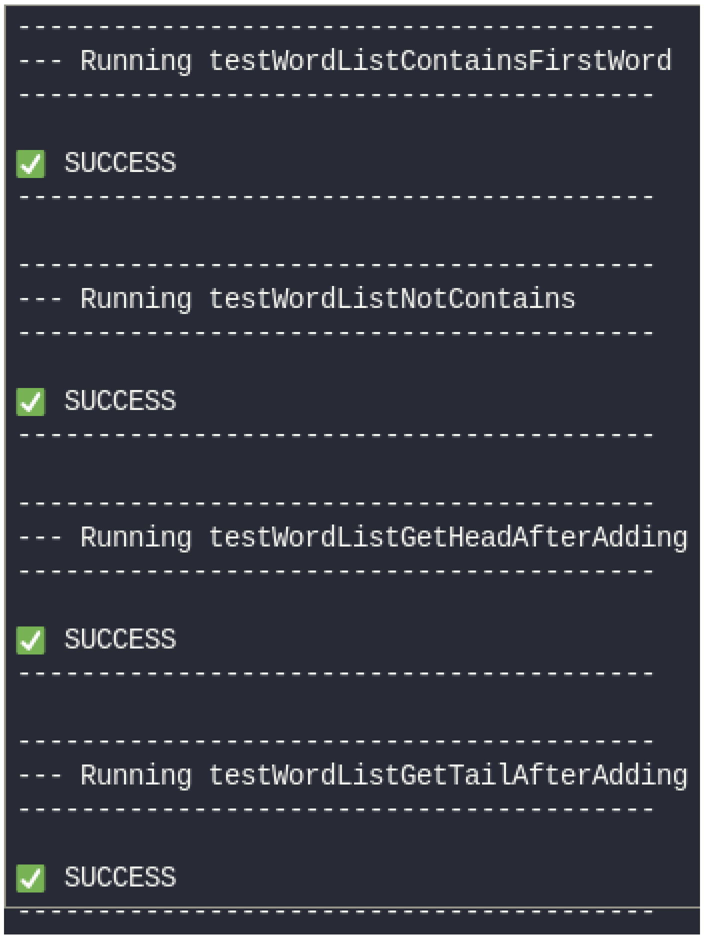
```
 src   main !11 ?8  ./TestWordList
----------------------------------------
--- Running testWordListCreate
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListDestroy
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListGetHead
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListSetHead
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListGetTail
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListSetTail
----------------------------------------

✅ SUCCESS
----------------------------------------
```



```
----------------------------------------
--- Running testWordListNullSetHead
----------------------------------------
Error: Cannot set head of NULL WordList

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListNullSetTail
----------------------------------------
Error: Cannot set head of NULL WordList

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListSetNullHead
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListSetNullTail
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListEmptyContains
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListAddWord
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListAddThreeWord
----------------------------------------
I1: text: word1, count: 1
I2: text: word2, count: 1
I3: text: word3, count: 1

L1: text: word1, count: 1
L2: text: word2, count: 1
L3: text: word3, count: 1

✅ SUCCESS
----------------------------------------
```

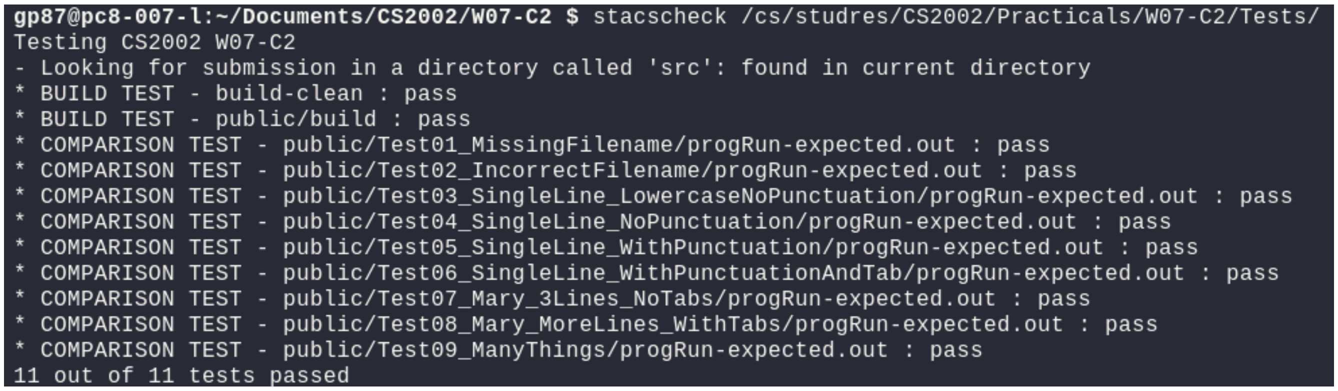


```
----------------------------------------
--- Running testWordListAddTwoDuplicateWords
----------------------------------------
I1: text: word1, count: 1
I2: text: word2, count: 1
I3: text: word2, count: 1
I4: text: word2, count: 1

L1: text: word1, count: 1
L2: text: word2, count: 1
L3: text: word2, count: 2
L4: text: word2, count: 3

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListAddOverflowCount
----------------------------------------
Error: Max Count for Word "george" reached
Word "george" in list exceeded maximum count

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListAddDestroyWords
----------------------------------------

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListNullAddWord
----------------------------------------
Error: Cannot add Word to a NULL WordList

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListAddNullWord
----------------------------------------
Error: Word is NULL

✅ SUCCESS
----------------------------------------

----------------------------------------
--- Running testWordListContains
----------------------------------------

✅ SUCCESS
----------------------------------------
```



```
----------------------------------------
--- Running testWordListContainsFirstWord
----------------------------------------

✅ SUCCESS
----------------------------------------


----------------------------------------
--- Running testWordListNotContains
----------------------------------------

✅ SUCCESS
----------------------------------------


----------------------------------------
--- Running testWordListGetHeadAfterAdding
----------------------------------------

✅ SUCCESS
----------------------------------------


----------------------------------------
--- Running testWordListGetTailAfterAdding
----------------------------------------

✅ SUCCESS
----------------------------------------
```

Here is the output of stacscheck. These tests show that the program successfully deals with various examples, as well as issues where the filename is not provided, and where the file is invalid. It also shows that my code can handle many different scenarios, from upper case, mixed case, to odd symbols, as well as dealing with many of the same words and many different words.



```
gp87@pc8-007-l:~/Documents/CS2002/W07-C2 $ stacscheck /cs/studres/CS2002/Practicals/W07-C2/Tests/
Testing CS2002 W07-C2
- Looking for submission in a directory called 'src': found in current directory
* BUILD TEST - build-clean : pass
* BUILD TEST - public/build : pass
* COMPARISON TEST - public/Test01_MissingFilename/progRun-expected.out : pass
* COMPARISON TEST - public/Test02_IncorrectFilename/progRun-expected.out : pass
* COMPARISON TEST - public/Test03_SingleLine_LowercaseNoPunctuation/progRun-expected.out : pass
* COMPARISON TEST - public/Test04_SingleLine_NoPunctuation/progRun-expected.out : pass
* COMPARISON TEST - public/Test05_SingleLine_WithPunctuation/progRun-expected.out : pass
* COMPARISON TEST - public/Test06_SingleLine_WithPunctuationAndTab/progRun-expected.out : pass
* COMPARISON TEST - public/Test07_Mary_3Lines_NoTabs/progRun-expected.out : pass
* COMPARISON TEST - public/Test08_Mary_MoreLines_WithTabs/progRun-expected.out : pass
* COMPARISON TEST - public/Test09_ManyThings/progRun-expected.out : pass
11 out of 11 tests passed
```