Homework 2: Debugging Functions

Released Friday 9/26/16 - 5:00pm

Due Friday 9/9/16 - 11:59pm

Goals:

- Further develop your debugging skills, and knowledge of GDB

Task: Debugging Functions

Navigate to Vocareum and select HW2, you will find a file called "**functions.c**". This file contains many different functions with different bugs in each. Included in the file is a description of each function and a sample output for each. Your task is to fix all of the bugs so that the functions all work properly.

Given:

functions.c - File containing functions with bugs.
functions.h - header file for **functions.c**
testing_tool.c - helps to check if the program works properly
testing_tool.org:
tool you may use to see how the functions should run if done properly
Makefile:
compiles source files to the executable files as **testing_tool** and **test_functions**

Arrays refresher:

How to initialize an array in C	
int array[5];	➢ Initializes an integer array of size 5
int array[] = {7 , 6 , 2 , 1};	➢ Initializes an integer array of size 4
char string[7];	➢ Initializes a char array of size 7
char string[] = “Hello\n”;	➢ Initializes a char array of size 7 ➔ Note: Remember that all strings in C end with a null terminator, ‘\0’

How to access values
int array[] = {86, 3, 7, 12}; int x = array[2]; ------------------------------> x now stores the value 7;
char string[] = “barrage” char c = string[6]; ---------------------------> c now stores ‘e’ c = string[7]; ---------------------------------> c now stores ‘\0’

Steps:

1. Open “**functions.c**” and read through the descriptions of each function
2. Run **make** to compile **testing_tool.c** and **functions.c** to the executable file **testing_tool**
3. Use **gdb** to debug the executable program find the errors in each of the functions and **fix** the bugs you found by modifying “**functions.c**”

➔ Note: Before looking at the functions, think about refreshing yourself on some of the binary operators such as “&, |, >>, <<”. You will not be allowed to add any other libraries from the C standard library to the file.

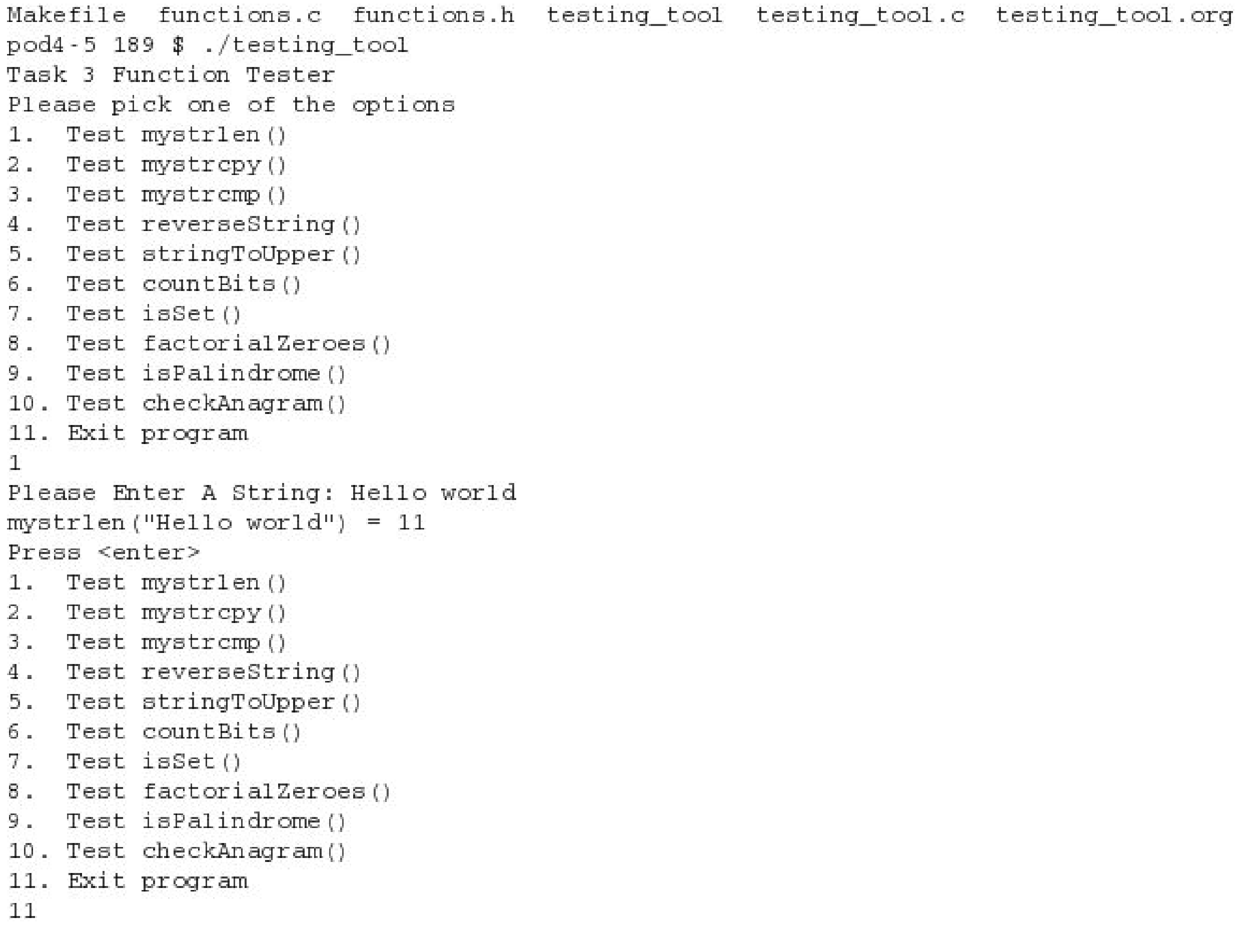
4. Use “**testing_tool**” and “**testing_tool.org**” to test the functions.

➔ Note: Remember to test all functions carefully and think about all possible cases. There is at least one bug per function.

5. On Test bubbleSort(), if you find the numbers to be doing correctly, try with a larger array size.

Testing_tool Demo:

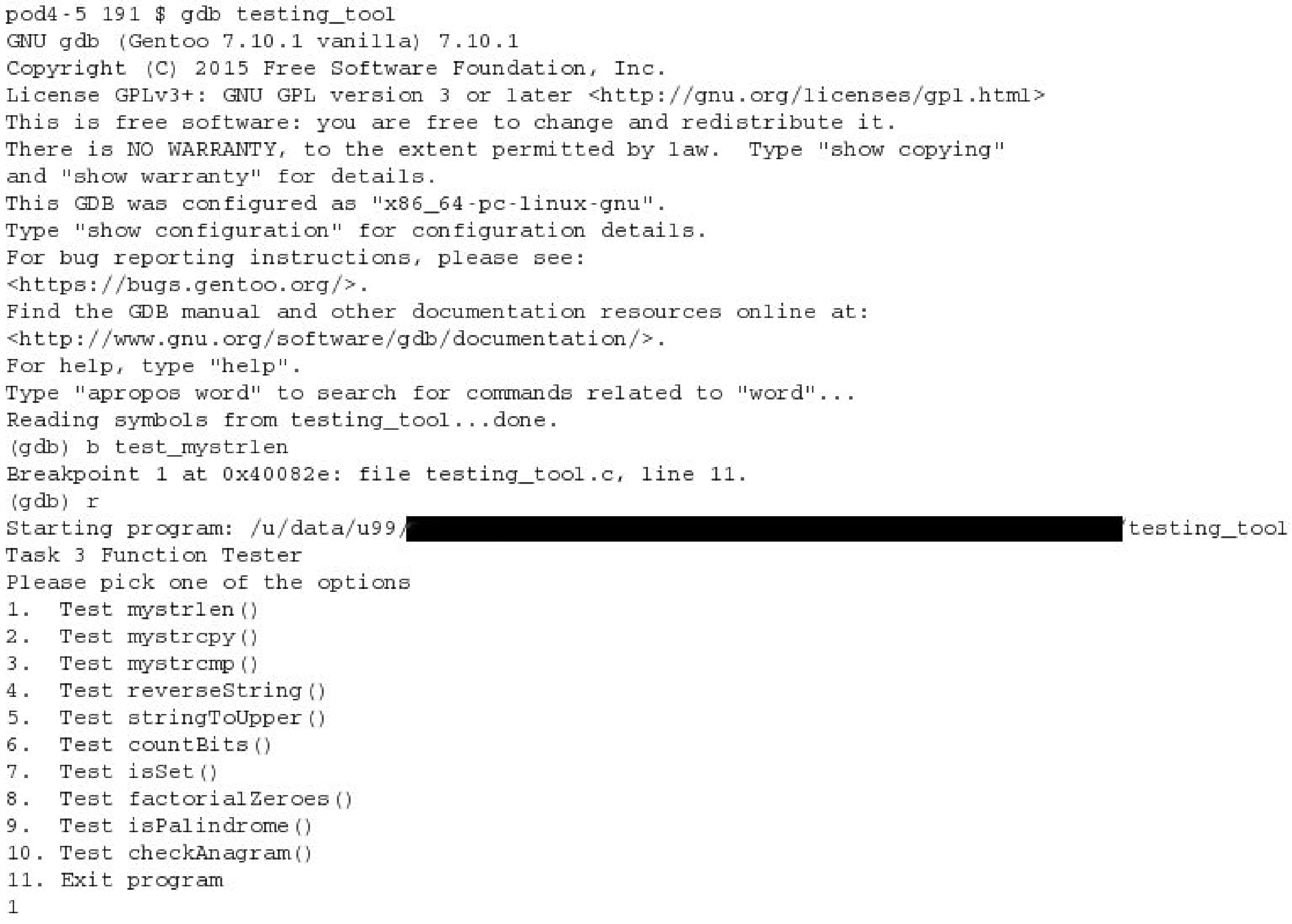
In order to test that your functions are working properly, you can use the “**testing_tool**” file to test your functions with inputs. Type “**make**” to compile your functions along with “**testing_tool.c**”. You may then type “**./testing_tool**” to run the program. You will be presented with a menu of all the different functions to test as shown below. Pick a function to test, and the program will ask you to enter inputs to the function.



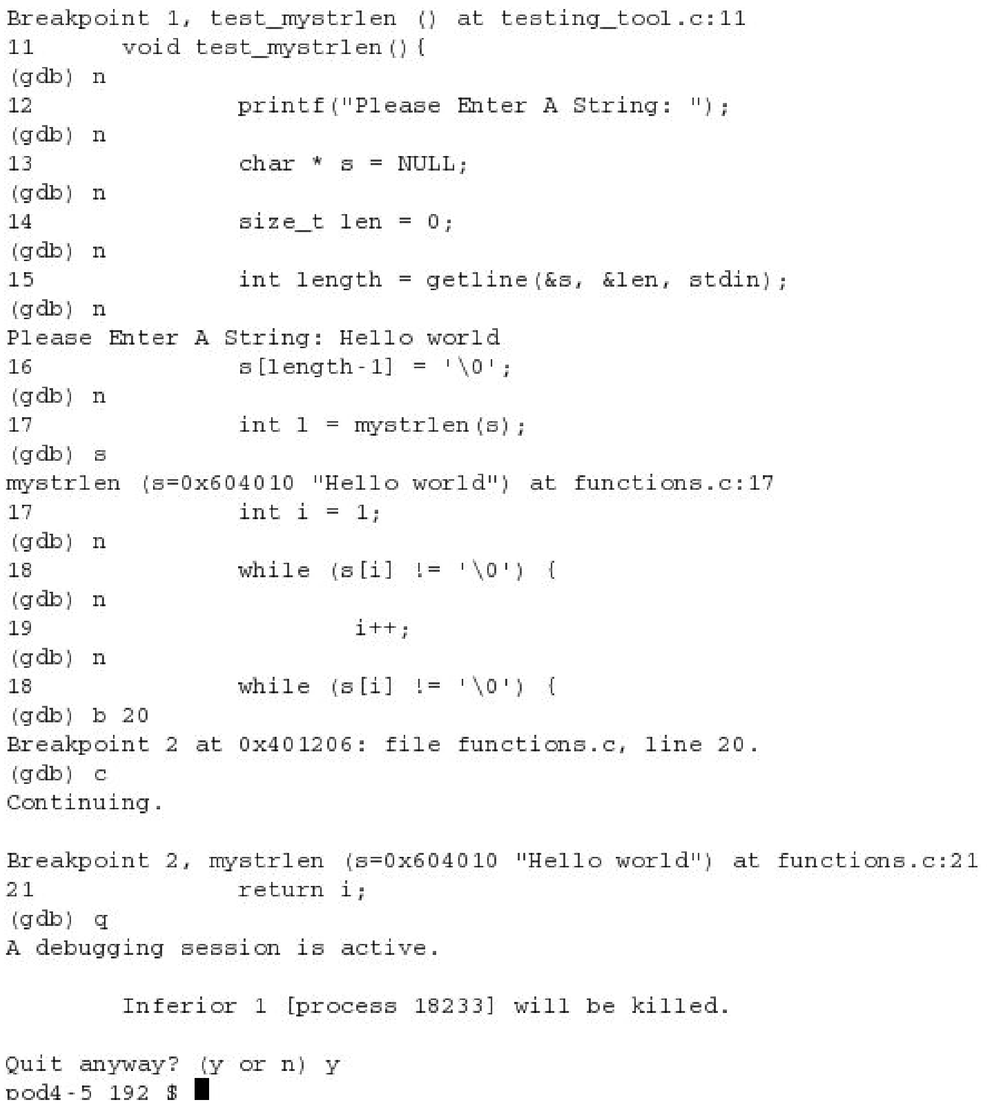
```
Makefile  functions.c  functions.h  testing_tool  testing_tool.c  testing_tool.org
pod4-5 189 $ ./testing_tool
Task 3 Function Tester
Please pick one of the options
1.  Test mystrlen()
2.  Test mystrcpy()
3.  Test mystrcmp()
4.  Test reverseString()
5.  Test stringToUpper()
6.  Test countBits()
7.  Test isSet()
8.  Test factorialZeroes()
9.  Test isPalindrome()
10. Test checkAnagram()
11. Exit program
1
Please Enter A String: Hello world
mystrlen("Hello world") = 11
Press <enter>
1.  Test mystrlen()
2.  Test mystrcpy()
3.  Test mystrcmp()
4.  Test reverseString()
5.  Test stringToUpper()
6.  Test countBits()
7.  Test isSet()
8.  Test factorialZeroes()
9.  Test isPalindrome()
10. Test checkAnagram()
11. Exit program
11
```

You may use **gdb** along with testing_tool.c. Type “**gdb testing_tool**” to open up gdb. Set a breakpoint using “**b**” or “**break**” at whichever function you would like to test. You can then run through the function line by line.

.



```
pod4-5 191 $ gdb testing_tool
GNU gdb (Gentoo 7.10.1 vanilla) 7.10.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from testing_tool...done.
(gdb) b test_mystrlen
Breakpoint 1 at 0x40082e: file testing_tool.c, line 11.
(gdb) r
Starting program: /u/data/u99/[redacted]/testing_tool
Task 3 Function Tester
Please pick one of the options
1.  Test mystrlen()
2.  Test mystrcpy()
3.  Test mystrcmp()
4.  Test reverseString()
5.  Test stringToUpper()
6.  Test countBits()
7.  Test isSet()
8.  Test factorialZeroes()
9.  Test isPalindrome()
10. Test checkAnagram()
11. Exit program
1
```



```
Breakpoint 1, test_mystrlen () at testing_tool.c:11
11      void test_mystrlen(){
(gdb) n
12              printf("Please Enter A String: ");
(gdb) n
13              char * s = NULL;
(gdb) n
14              size_t len = 0;
(gdb) n
15              int length = getline(&s, &len, stdin);
(gdb) n
Please Enter A String: Hello world
16              s[length-1] = '\0';
(gdb) n
17              int l = mystrlen(s);
(gdb) s
mystrlen (s=0x604010 "Hello world") at functions.c:17
17              int i = 1;
(gdb) n
18              while (s[i] != '\0') {
(gdb) n
19                      i++;
(gdb) n
18              while (s[i] != '\0') {
(gdb) b 20
Breakpoint 2 at 0x401206: file functions.c, line 20.
(gdb) c
Continuing.

Breakpoint 2, mystrlen (s=0x604010 "Hello world") at functions.c:21
21              return i;
(gdb) q
A debugging session is active.

        Inferior 1 [process 18233] will be killed.

Quit anyway? (y or n) y
pod4-5 192 $
```

Submission:

Make edits to **function.c** to make it run without any errors, and submit. The Vocareum Terminal will be activated for this assignment. You can either work on Vocareum or offline, but to submit **function.c** must be in your **work** dir.