Rhea Mae Edwards
Student ID #932-389-303
CS 362 Midterm (Spring 2016)

void myqsort(int a[], unsigned int size)
Quick sort of integer arrays in C (pretty fast sorting method)

Describe or write (pseudo)code for the following two goals:

1) Automatically generate tests for this function

In order to automatically generate tests for a function that implements a quick sort of integer arrays in C, a programmer can initially write a unit test testing if the function sorts correctly.

For example, possibly just running the function with an input of an array of three integers such as 0, 1, and 2, and:

- One way, giving that array in an already sorted order, making sure it outputs the same already sorted array afterwards.
- Another way, is scrabbling the array of the three integers (not already previously sorted), and checking that the function sorts those integers correctly, outputting 0, 1, and then 2.

If the unit tests checks that the output holds true (as in a sorted array), then the function of myqsort has been tested for correct functionality being corrects. If the unit tests outputs finding out incorrect functionality of myqsort, then a programmer will/could continue testing the function of myqsort with more yet different type of testing unit tests, in order to keep automatically generating tests for the function. Such as certain individual or section of statements within the code, like how the function actually moves an element within the array in order to sort it.

2) Check if the results are correct. In particular, is the following check sufficient? If not, how should it be improved?

The following check does contain the size of the array, keeps record of the initially inputted array, checks if one element is smaller than its following element, and understands that if the initially inputted array contains an integer, the array outputted by the function myqsort will also contain that certain integer, along with the number of times it does show up (i.e. once, twice, etc.), making sure it's the same array that was inputted outputted.

But the entirety of the check is not sufficient when it comes to checking if the outputted array is completely sorted after going through the function. Because it will only check adjacent elements of the outputted array, to see if the condition stated fits. It does not check if the entire array is truly sorted. For example, the element of a[1] can be greater than the element of a[4] where it is supposed to be less than a[4]. Also the check doesn't recheck that the outputted array is the same size as the originally array that it took account for beforehand. In order to improve this check, it would have to do these functionalities: check the entire array that it is truly sorted, and making sure that the size of the outputted array is the same size as the originally array.

3) Finally, how would you check that the test generation and check for correctness are sufficiently strong to test quick sorting?

“**Generation** of test based on coverage means producing a test suite to achieve a certain level of coverage”

“**Test Suite:** a *set* of executions of a program, grouped together”

A way to check that the test generation and correctness are sufficiently strong to test quick sorting is to run the tests using gcov to collect this coverage data. gcov being a tool that comes with GCC for collecting and analyzing coverage.

A programmer can compile gcov along with additional items, and when the executable is ran with the tests for myqsort function, it will produce files (gcda files) that record how often each line ran (making sure that the program for the myqsort function along with the tests, have been ran first before checking for any coverage). Then further observation and analysis of this data collected by the programmer can be done to further understand the sufficiently of the tests, checking that they are sufficiently strong to test quick sorting.