Test Plan - Garmin Simple Compression

**Prepared by:**

Lesley Linge

November 14th, 2020

Version 1.0

**1.0 Introduction**

The product being tested is a simple function that will compress a series of 7-bit integers. The function is called with 2 arguments; a pointer to the data buffer and number of bytes to compress. The function modifies data in the buffer and returns the size of the modified (compressed) buffer.

**2.0 OBJECTIVES AND TASKS**

**2.1 Objectives**

This Test Plan document is meant to display the skills and competencies of the applicant Lesley Linge for the Embedded Test Software Engineer position with the Fitness Division of Garmin.

**2.2 Tasks**

1. Test Plan
2. Test Cases
3. Run Report & Analysis
4. Source code
5. Submission

**3.0 SCOPE**

**General:** The code in question is stand alone and relatively simple so the testing will be limited to unit testing. The testing will include happy path functional testing and boundary testing. Performance testing could be added if needed.

**Tactics:** Inputs will be matched with expected outputs**.** A reverse compression algorithm will be used on the compressed datato verify that it matches the input.

**4.0 TESTING STRATEGY**

One way of doing this is to use known inputs and their corresponding expected outputs. A more thorough way would be to implement a decompression function that would modify the data buffer back to the original form and compare the buffer before and after. Another way would be generation of random data with corresponding expected output. A combination of these will be used. Boundaries and special cases should be defined and create their respective test cases.

For ease of scripting and generation of data a python test using ctypes will send and receive data the the c function to test it.

**4.1 Unit Testing**

**Definition:** The data will be randomly generated with a parameter for the amount of test data to be generated so the amount of testing is mostly limited by the time for testing. The current number of random tests is set to 1000 and completes in a few seconds. The test has build in reporting that goes to the console (this can be easily changed to log) and the simplest form of logging is to pipe the output to a file.

**Participants:** Lesley Linge

**Methodology:** Lesley Linge will create and maintain the test script. The script can be added to the CI (such as Jenkins) and used for regression. The most important is boundary testing. The implementation is rather limited as can be expected in an embedded environment where resources are chosen carefully and spec’d to be sufficient but not superfluous. The use of only 7 bits means that only numbers 0-127 are available and the max a number may repeat is 127 times. If support for more than 127 repeating numbers if necessary the code could be modified to count a max of 127 repeating numbers and then start counting from 0 for any further repeats.Due to the lack of any input checking or error handling in the function many of the boundary tests are expected to fail on out of bounds situations. These are the test cases:

1. Randomized input data with corresponding expected data and decompression vs original comparison.
2. 0 size data
3. data greater than 127 bytes
4. nums greater than 127
5. 0 repeated 127 times
6. 127 repeated 127 times
7. data with repeating 255 times
8. data with repeating 128 times
9. data with 255 repeating 255 times

**4.2 System and Integration Testing**

**Definition:** Out of scope as there is not anything to integrate it with.

**Participants:**

**Methodology:** Out of scope.

**4.3 Performance and Stress Testing**

**Definition:** A baseline on well defined hardware tracked over time for performance degredation.

**Participants:** Lesley

**Methodology:** The script can be used to generate and save a dataset and the script can be easily expanded upon to load it. A baseline performance can be recorded on specific hardware and the same test can be run periodically to track for any performance degradation over time.

**4.4 User Acceptance Testing**

**Definition:** Out of scope as there is no user defined and the purpose and **requirements** of the function are not fleshed out.

**Participants:**

**Methodology:** Out of scope.

**4.5 Batch Testing**

**Definition:** Out of scope the function is stateless.

**Participants:**

**Methodology:** Out of scope.

**4.6 Automated Regression Testing**

**Definition:** A CI such as Jenkins should be set up to automate testing through use of the unit test triggered by code changes.

**4.7 Beta Testing**

**Definition:** Out of scope as there is no user defined and the purpose and requirements of the function are not fleshed out. There is also no current release schedule for the function.

**Participants:**

**Methodology:** Out of scope.

**5.0 HARDWARE REQUIREMENTS**

A desktop computer or laptop is sufficient

**6.0 ENVIRONMENT REQUIREMENTS**

The unit test relies on python 3, numpy, and ctypes.

**7.0 TEST SCHEDULE**

The Test Plan, Test Cases, Run Report & Analysis and source code should be provided to Garmin by Monday, November 16, 2020 at 11:59 pm MST.

**8.0 CONTROL PROCEDURES**

**Problem Reporting**

The procedures to be followed when an incident, test failure or crash should be documented here. What is the logging required, who needs to be informed? Issues will be reported here: <https://github.com/ldlinge/SimpleCompressionTest/issues>

**Change Requests**

Lesley Linge will sign off on changes to this document or the requirements.

**9.0 FEATURES TO BE TESTED**

The data compression function shall be tested.

**10.0 FEATURES NOT TO BE TESTED**

**11.0 RESOURCES/ROLES & RESPONSIBILITIES**

Lesley Linge will design the test plan (this document), implement test cases, and include a run report and source code.

**12.0 SCHEDULES**

**Major Deliverables:**

* Test Plan
* Test Cases
* Test Summary Report
* Source code = <https://github.com/ldlinge/SimpleCompressionTest/>

These will all be made available to Garmin by Monday, November 16, 2020 at 11:59 pm MST.

**13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)**

Garmin Fitness division QA

**14.0 DEPENDENCIES**

**15.0 RISKS/ASSUMPTIONS**

Identify the high-risk assumptions of the test plan. Specify contingency plans for each (**For Example,** delay in delivery of test items might require increased night shift scheduling to meet the delivery date).

**16.0 TOOLS**

Visual Studio, Jenkins, Python. Bugs will be tracked at <https://github.com/ldlinge/SimpleCompressionTest/issues>

**17.0 APPROVALS**

Specify the names and titles of all the persons who must approve this plan. Provide space for the signatures and dates.

Name (In Capital Letters) Signature Date:

1.

2.

3.

4.