Final Report Submission

Software Testing

# Submitted to**:** Dr. Ahmed Sobeih

Eng. Ahmed Bahgat

Eng. Mohamed Adel

Submitted by:

Name: Ahmed Saleh Email: [abosalaah96@gmail.com](mailto:abosalaah96@gmail.com) Name: Mohamed Hamada Email: [homshamada@gmail.com](mailto:homshamada@gmail.com) Name: Mina Magdy Email: [mina\_mego5@yahoo.com](mailto:mina_mego5@yahoo.com) Name: Omar Osama Email: [omarosamasobeih@yahoo.com](mailto:omarosamasobeih@yahoo.com)

**Table of Contents**

1. Tool Overview
2. How to build the tool
3. Logic Coverage
4. Other Coverage Criteria
5. Statistics
6. References
7. Workload

Tool Overview:

We used tool named “CppUTest” and it is a C /C++ based unit xUnit test framework for unit testing and for test-driving your code. It is written in C++ but is used in C and C++ projects and frequently used in embedded systems but it works for any C/C++ project.

And the tool has many assertions like:

1. CHECK\_EQUAL(expected, actual) - checks for equality between entities using “==” operator. So, to use it we need to have this operator in the datatype we are comparing.
2. STRCMP\_EQUAL(expected, actual) - compares two “ char \* “ arrays using “strcmp” function
3. POINTERS\_EQUAL(expected, actual) - compares two pointers and check if they are equal and equal here means they have the same address.
4. CHECK\_THROWS(expected\_exception, expression) - checks if expression throws expected\_exception (e.g. std::exception). CHECK\_THROWS is only available if CppUTest is built with the Standard C++ Library (default).
5. FAIL(text) - always fails

How to build it

* Windows Visual Studio
* Create Visual Studio Project
* Project Properties -> C/C++ -> General ->
* Additional Include Directories -> path of include folder in the library folder
* Project Properties -> Linker -> General -> Additional Library Directories -> path of lib folder in the library folder
* Project Properties -> Linker -> Input -> Additional Dependencies -> Add “CppUTestd.lib”
* Linux : using this command
* apt-get install cpputest
* MacOSX : using this command
* brew install cpputest

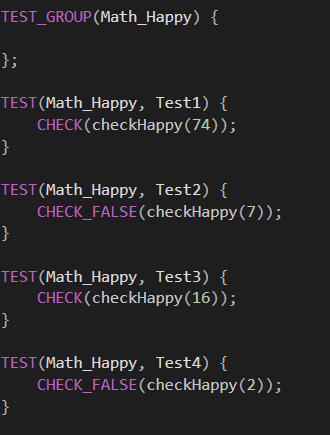
Logic Coverage:

1. First example: testing math function indicates happy numbers

Happy number in his definition that has 2 or more features (even, has perfect square root or lucky i.e. all digits are either ‘4’ or ‘7’).



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | isLucky | isEven | hasSqrt | P | P(isLucky) | P(isEven) | P(hasSqrt) | Input |
| 1 | True | True | True | True | False | False | False | 4 |
| 2 | True | True | False | True | True | True | False | 74 |
| 3 | True | False | True | True | True | False | True | - |
| 4 | True | False | False | False | False | True | True | 7 |
| 5 | False | True | True | True | False | True | True | 16 |
| 6 | False | True | False | False | True | False | True | 2 |
| 7 | False | False | True | False | True | True | False | 81 |
| 8 | False | False | False | False | False | False | False | 3 |



1. Combinatorial Clause Coverage (COC):

Infeasible to force the combination where “isLucky” is true, “isEven” is false and “hasSqrt” is true

1. Restricted Active Clause Coverage (RACC):

* Test Pair that satisfy RACC with respect to clause “isLucky”: (2,6)
* Test Pair that satisfy RACC with respect to clause “isEven”: (4,5)
* Test Pair that satisfy RACC with respect to clause “hasSqrt”: (5,6)

1. Restricted Inactive Clause Coverage (RICC):

Infeasible to force predicate to evaluate with true while “isEven” is major

1. Correlated Active Clause Coverage (CACC):

* Test Pair that satisfy CACC with respect to clause “isLucky”: (2,6)
* Test Pair that satisfy CACC with respect to clause “isEven”: (4,5)
* Test Pair that satisfy CACC with respect to clause “hasSqrt”: (5,6)

1. General Active Clause Coverage (GACC):

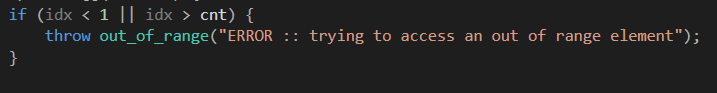
* Test Pair that satisfy GACC with respect to clause “isLucky”: (2,6)
* Test Pair that satisfy GACC with respect to clause “isEven”: (4,5)
* Test Pair that satisfy GACC with respect to clause “hasSqrt”: (5,6)

1. General Inactive Clause Coverage (GICC):

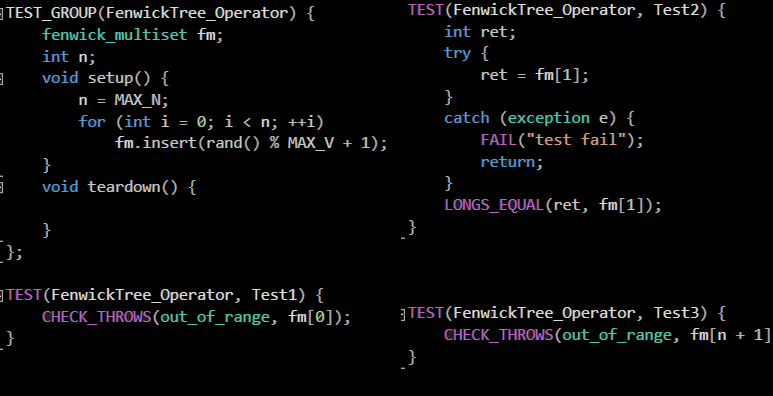
Infeasible to force predicate to evaluate with true while “isEven” is major

1. Clause Coverage (CC):

* Test Pair that satisfy CC with respect to clause “isLucky”: (1, 8)
* Test Pair that satisfy CC with respect to clause “isEven”: (1, 8)
* Test Pair that satisfy CC with respect to clause “hasSqrt”: (1, 8)

1. Predicate Coverage (PC): Test Pair that satisfy predicate coverage: (5, 6)
2. Second Example: testing operator[] in multiset implemented using Fenwick tree

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | idx < 1 | idx > cnt | P | P(idx<1) | P(idx>cnt) | idx |
| 1 | True | True | True | False | False | - |
| 2 | True | False | True | True | False | 0 |
| 3 | False | True | True | False | True | cnt + 1 |
| 4 | False | False | False | True | True | 1 |



1. Combinatorial Clause Coverage (COC):

Infeasible requirement

1. Restricted Active Clause Coverage (RACC):

* Test Pair that satisfy RACC with respect to clause “idx < 1”: (2,4)
* Test Pair that satisfy RACC with respect to clause “idx > cnt”: (3, 4)

1. Restricted Inactive Clause Coverage (RICC):

Infeasible requirement

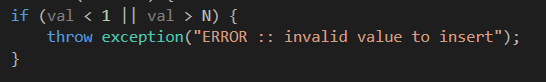
1. Clause Coverage (CC):

* Test Pair that satisfy CC with respect to clause “idx < 1”: (2,3)
* Test Pair that satisfy CC with respect to clause “idx > cnt”: (2, 3)

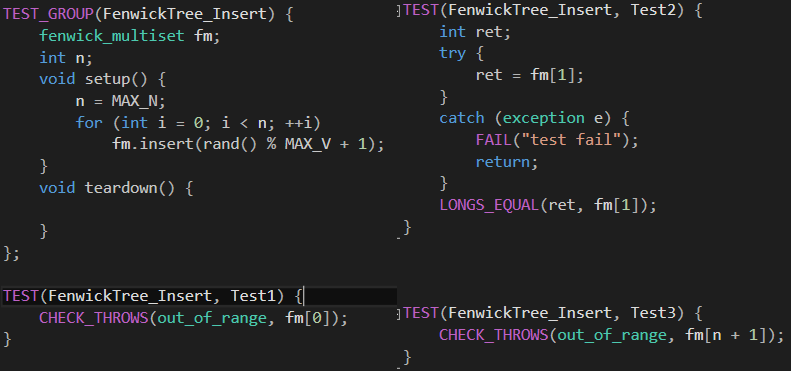
1. Predicate Coverage (PC):

Test Pair that satisfy predicate coverage: (3, 4)

1. Third Example: testing insert function in multiset implemented using Fenwick tree.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | val < 1 | val > N | P | P(val<1) | P(val>N) | val |
| 1 | True | True | True | False | False | - |
| 2 | True | False | True | True | False | 0 |
| 3 | False | True | True | False | True | N + 1 |
| 4 | False | False | False | True | True | 1 |



1. Combinatorial Clause Coverage (COC):

Infeasible requirement

1. Restricted Active Clause Coverage (RACC):

* Test Pair that satisfy RACC with respect to clause “val < 1”: (2,4)
* Test Pair that satisfy RACC with respect to clause “val > N”: (3, 4)

1. Restricted Inactive Clause Coverage (RICC):

Infeasible requirement

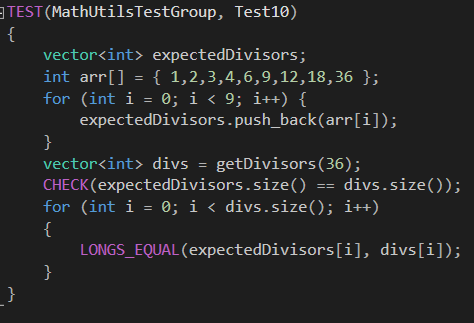
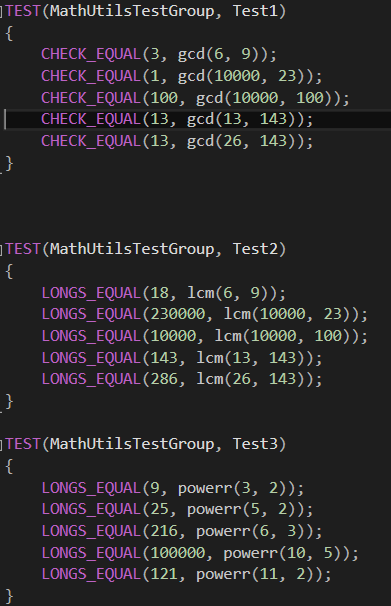
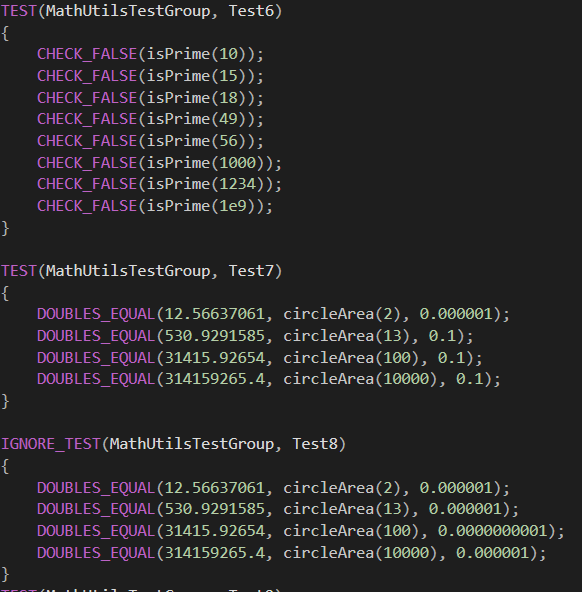
1. Clause Coverage (CC):

* Test Pair that satisfy CC with respect to clause “val < 1”: (2,3)
* Test Pair that satisfy CC with respect to clause “val > N”: (2, 3)

1. Predicate Coverage (PC):

Test Pair that satisfy predicate coverage: (2, 4)

Other Coverage Criteria:



Statistics:

Checks = 102

Tests = 62

60 Tests ran

2 Tests ignored

0 Tests filtered out

References:

<https://cpputest.github.io/>

<https://cpputest.github.io/manual.html>

Work Load:

|  |  |
| --- | --- |
| Name | Task |
| Omar Osama | Logic Coverage |
| Mina Magdy | Logic Coverage |
| Mohmed Hamada | Another Coverages & Tool administration |
| Ahmed Saleh | Another Coverages & Tool administration |