# Submission Report

* Submission generated at 20250822\_163314
* Machine info: Darwin segfault.local 24.6.0 Darwin Kernel Version 24.6.0: Mon Jul 14 11:30:29 PDT 2025; root:xnu-11417.140.69~1/RELEASE\_ARM64\_T6000 arm64

## Build Output

/Applications/Xcode.app/Contents/Developer/usr/bin/make BUILD=debug  
mkdir -p build/debug  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address -c src/lab.c -o build/debug/lab.c.o  
mkdir -p build/debug  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address -c src/main.c -o build/debug/main.c.o  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address build/debug/lab.c.o build/debug/main.c.o -o build/debug/myapp\_d -fsanitize=address  
/Applications/Xcode.app/Contents/Developer/usr/bin/make BUILD=release  
mkdir -p build/release  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -fstrict-flex-arrays=3 -fstack-protector-strong -Werror=format-security -Werror=implicit -Werror=incompatible-pointer-types -Werror=int-conversion -c src/lab.c -o build/release/lab.c.o  
mkdir -p build/release  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -fstrict-flex-arrays=3 -fstack-protector-strong -Werror=format-security -Werror=implicit -Werror=incompatible-pointer-types -Werror=int-conversion -c src/main.c -o build/release/main.c.o  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -fstrict-flex-arrays=3 -fstack-protector-strong -Werror=format-security -Werror=implicit -Werror=incompatible-pointer-types -Werror=int-conversion build/release/lab.c.o build/release/main.c.o -o build/release/myapp   
/Applications/Xcode.app/Contents/Developer/usr/bin/make BUILD=debug-test  
mkdir -p build/debug-test  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address -c src/lab.c -o build/debug-test/lab.c.o  
mkdir -p build/debug-test  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address -c src/main.c -o build/debug-test/main.c.o  
mkdir -p build/debug-test/harness/  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address -c tests/harness/unity.c -o build/debug-test/harness/unity.c.o  
mkdir -p build/debug-test/  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address -c tests/lab-test.c -o build/debug-test/lab-test.c.o  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address build/debug-test/lab.c.o build/debug-test/main.c.o build/debug-test/harness/unity.c.o build/debug-test/lab-test.c.o -o build/debug-test/myapp\_td -fsanitize=address  
/Applications/Xcode.app/Contents/Developer/usr/bin/make BUILD=test  
mkdir -p build/tests  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage -c src/lab.c -o build/tests/lab.c.o  
mkdir -p build/tests  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage -c src/main.c -o build/tests/main.c.o  
mkdir -p build/tests/harness/  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage -c tests/harness/unity.c -o build/tests/harness/unity.c.o  
mkdir -p build/tests/  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage -c tests/lab-test.c -o build/tests/lab-test.c.o  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage build/tests/lab.c.o build/tests/main.c.o build/tests/harness/unity.c.o build/tests/lab-test.c.o -o build/tests/myapp\_t -fprofile-arcs -ftest-coverage  
All builds completed: debug, release, and test.

## Coverage Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:44:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:45:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:46:test\_subtract:PASS  
  
-----------------------  
3 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:44:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:45:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:46:test\_subtract:PASS  
  
-----------------------  
3 Tests 0 Failures 0 Ignored   
OK  
mkdir -p ./build/report/html  
mkdir -p ./build/report/txt  
gcovr -r . --html --html-details --exclude-directories build/tests/harness --exclude '.\*main\.c$' --exclude '.\*test\.c$' -o ./build/report/html/coverage\_report.html  
(INFO) Reading coverage data...  
(INFO) Writing coverage report...  
gcovr -r . --txt --exclude-directories build/tests/harness --exclude '.\*main\.c$' --exclude '.\*test\.c$'  
(INFO) Reading coverage data...  
(INFO) Writing coverage report...  
------------------------------------------------------------------------------  
 GCC Code Coverage Report  
Directory: .  
------------------------------------------------------------------------------  
File Lines Exec Cover Missing  
------------------------------------------------------------------------------  
src/lab.c 13 13 100%  
------------------------------------------------------------------------------  
TOTAL 13 13 100%  
------------------------------------------------------------------------------

## Address Sanitizer Report

==3032==AddressSanitizer: detect\_leaks is not supported on this platform.  
/bin/bash: line 1: 3032 Abort trap: 6 ASAN\_OPTIONS="detect\_leaks=1" ./build/debug-test/myapp\_td  
make: \*\*\* [leak-test] Error 134

## Source Files

### lab.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
int add(int a, int b) {  
 return a + b;  
}  
  
int subtract(int a, int b) {  
 return a - b;  
}  
  
char \*get\_greeting(const char \*restrict name)  
{  
 if (name == NULL)  
 {  
 return NULL;  
 }  
  
 // Allocate memory for the greeting message  
 int length = snprintf(NULL, 0, "Hello, %s!", name);  
 if (length < 0) // GCOVR\_EXCL\_START  
 {  
 return NULL; // snprintf failed  
 } // GCOVR\_EXCL\_STOP  
  
 //Casting is safe here because we know length is non-negative  
 size\_t alloc\_size = (size\_t) length + 1; // +1 for the null terminator  
 char \*greeting = malloc( alloc\_size);  
  
  
 if (greeting == NULL) // GCOVR\_EXCL\_START  
 {  
 return NULL; // Memory allocation failed  
 } // GCOVR\_EXCL\_STOP  
  
  
 // Create the greeting message  
 snprintf(greeting, alloc\_size, "Hello, %s!", name);  
  
 return greeting;  
}

### lab.h

#ifndef LAB\_H  
#define LAB\_H  
  
/\*\* \* @brief Returns a greeting message.  
 \*  
 \* This function returns a string that contains a greeting message.  
 \* The string is allocated with malloc and should be freed by the caller.  
 \* @param name The name to include in the greeting.  
 \* @return A greeting string.  
 \*/  
char\* get\_greeting(const char\* restrict name);  
  
/\*\* \* @brief Adds two integers.  
 \*  
 \* This function adds two integers and returns the result.  
 \* @param a The first integer.  
 \* @param b The second integer.  
 \* @return The sum of a and b.  
 \*/  
int add(int a, int b);  
  
/\*\* \* @brief Subtracts two integers.  
 \*  
 \* This function subtracts the second integer from the first and returns the result.  
 \* @param a The first integer.  
 \* @param b The second integer.  
 \* @return The result of a - b.  
 \*/  
int subtract(int a, int b);  
  
  
  
#endif // LAB\_H

### main.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
#ifdef TEST  
#define main main\_exclude  
#endif  
  
  
  
int main(void)  
{  
 int result\_add = add(5, 3);  
 int result\_subtract = subtract(5, 3);  
 printf("Addition Result: %d\n", result\_add);  
 printf("Subtraction Result: %d\n", result\_subtract);  
 char \*greeting = get\_greeting("World");  
 if (greeting) {  
 printf("%s\n", greeting);  
 free(greeting); // Free the allocated memory for the greeting  
 } else {  
 printf("Failed to create greeting.\n");  
 }  
 return 0;  
}

## Test Files

### lab-test.c

#include <stdlib.h>  
#include <stdio.h>  
#include "harness/unity.h"  
#include "../src/lab.h"  
  
  
void setUp(void) {  
 printf("Setting up tests...\n");  
}  
  
void tearDown(void) {  
 printf("Tearing down tests...\n");  
}  
  
void test\_add(void) {  
 TEST\_ASSERT\_EQUAL(8, add(5, 3));  
 TEST\_ASSERT\_EQUAL(-2, add(-5, 3));  
 TEST\_ASSERT\_EQUAL(0, add(0, 0));  
}  
  
void test\_subtract(void) {  
 TEST\_ASSERT\_EQUAL(2, subtract(5, 3));  
 TEST\_ASSERT\_EQUAL(-8, subtract(-5, 3));  
 TEST\_ASSERT\_EQUAL(0, subtract(0, 0));  
}  
  
void test\_get\_greeting(void) {  
 char \*greeting = get\_greeting("Alice");  
 TEST\_ASSERT\_NOT\_NULL(greeting);  
 TEST\_ASSERT\_EQUAL\_STRING("Hello, Alice!", greeting);  
 free(greeting); // Free the allocated memory for the greeting  
  
 greeting = get\_greeting(NULL);  
 TEST\_ASSERT\_NULL(greeting);  
  
 greeting = get\_greeting("");  
 TEST\_ASSERT\_NOT\_NULL(greeting);  
 TEST\_ASSERT\_EQUAL\_STRING("Hello, !", greeting);  
 free(greeting);  
}  
  
int main(void) {  
 UNITY\_BEGIN();  
 RUN\_TEST(test\_get\_greeting);  
 RUN\_TEST(test\_add);  
 RUN\_TEST(test\_subtract);  
 return UNITY\_END();  
}

## README.md

# Makefile Project Template  
  
This is a simple Makefile project template that can be used to build, test, and  
debug C projects. It includes support for debug builds, sanitizers, and code  
coverage.  
  
## Test Harness  
  
This project uses the Unity Test Framework for unit testing. Refer to the  
[Unity Getting Started Guide](https://github.com/ThrowTheSwitch/Unity/blob/master/docs/UnityGettingStartedGuide.md) for more information on how to write and run tests.  
  
## Example Usage  
  
To build the project run:  
  
```bash  
make release

To run the executable:

./build/release/myapp

To run the unit tests:

make check

To see all the configurations, run make help

Usage: make [target]  
Available targets:  
 debug - Build the application in debug mode (default)  
 release - Build the application in release mode  
 test - Build the unit tests  
 all - Builds debug, release, and test targets  
 check - Run tests and check results  
 report - Generate coverage report after running tests  
 leak - Check for memory leaks in debug mode  
 clean - Remove build artifacts  
 print - Print build variables for MakeFile debugging  
 help - Show this help message

## VS Code Integration

This project is designed to work well with Visual Studio Code. Configurations for debugging the application and unit tests are provided. Read about how to use the debugger in the [VS Code documentation](https://code.visualstudio.com/docs/editor/debugging).

## Features

* Build targets for debug and release modes
* Support for Address Sanitizer (ASan)
* Code coverage support and report generation
* Simple structure for organizing source files and build artifacts

 ```