# Submission Report

* Submission generated at 09/01/2025 at 20:26:37
* Machine info: Linux pkrvmccyg1gnepe 6.11.0-1018-azure #18~24.04.1-Ubuntu SMP Sat Jun 28 04:46:03 UTC 2025 x86\_64 x86\_64 x86\_64 GNU/Linux

## Build Output

make BUILD=debug  
make[1]: Entering directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
mkdir -p build/debug  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address -c src/main.c -o build/debug/main.c.o  
mkdir -p build/debug  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address -c src/lab.c -o build/debug/lab.c.o  
cc -g -O0 -DDEBUG -fno-omit-frame-pointer -fsanitize=address build/debug/main.c.o build/debug/lab.c.o -o build/debug/myapp\_d -fsanitize=address  
make[1]: Leaving directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
make BUILD=release  
make[1]: Entering directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
mkdir -p build/release  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -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  
mkdir -p build/release  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -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  
cc -Wall -Wextra -O2 -fPIE -MMD -MP -Wformat -Wformat=2 -Wconversion -Wsign-conversion -Wimplicit-fallthrough -fstack-protector-strong -Werror=format-security -Werror=implicit -Werror=incompatible-pointer-types -Werror=int-conversion build/release/main.c.o build/release/lab.c.o -o build/release/myapp   
make[1]: Leaving directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
make BUILD=debug-test  
make[1]: Entering directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
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  
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 tests/lab-test.c -o build/debug-test/lab-test.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  
cc -g -O0 -DDEBUG -DTEST -fno-omit-frame-pointer -fsanitize=address build/debug-test/main.c.o build/debug-test/lab.c.o build/debug-test/lab-test.c.o build/debug-test/harness/unity.c.o -o build/debug-test/myapp\_td -fsanitize=address  
make[1]: Leaving directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
make BUILD=test  
make[1]: Entering directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
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  
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 tests/lab-test.c -o build/tests/lab-test.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  
cc -g -O0 -DTEST -fprofile-arcs -ftest-coverage build/tests/main.c.o build/tests/lab.c.o build/tests/lab-test.c.o build/tests/harness/unity.c.o -o build/tests/myapp\_t -fprofile-arcs -ftest-coverage  
make[1]: Leaving directory '/home/runner/work/makefile-project-starter/makefile-project-starter'  
All builds completed: debug, release, and test.

## Coverage Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:56:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:57:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:58:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:59:test\_multiply:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:60:test\_bad\_add:PASS  
  
-----------------------  
5 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:56:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:57:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:58:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:59:test\_multiply:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:60:test\_bad\_add:PASS  
  
-----------------------  
5 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 16 16 100%  
------------------------------------------------------------------------------  
TOTAL 16 16 100%  
------------------------------------------------------------------------------

## Address Sanitizer Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:56:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:57:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:58:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:59:test\_multiply:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:60:test\_bad\_add:PASS  
  
-----------------------  
5 Tests 0 Failures 0 Ignored   
OK

## Source File: 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;  
}  
  
int multiply(int a, int b) {  
 // Example of an intentional memory leak  
 // int \*leak = malloc(sizeof(int));  
 // \*leak = a \* b;  
 // leak is never freed -> memory leak  
  
 return a \* b;  
}  
  
int bad\_add(int a, int b) {  
 return a + b + 1; // intentionally incorrect  
}  
  
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;  
}

## Source File: 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);  
  
/\*\*   
 \* @brief Multiplies two integers.  
 \*  
 \* This function multiplies two integers and returns the product.  
 \* @param a The first integer.  
 \* @param b The second integer.  
 \* @return The product of a and b.  
 \*/  
int multiply(int a, int b);  
  
/\*\*   
 \* @brief Returns an incorrect sum of two integers.  
 \*  
 \* This function intentionally returns an incorrect sum  
 \* by adding an extra 1 to the result.  
 \* @param a The first integer.  
 \* @param b The second integer.  
 \* @return The incorrect sum of a and b.  
 \*/  
int bad\_add(int a, int b);  
  
#endif // LAB\_H

## Source File: main.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
#ifdef TEST  
#define main main\_exclude  
#endif  
  
int main(void)  
{  
 // // Intentional crash: dereferencing a NULL pointer  
 // int \*p = NULL;  
 // \*p = 42; // This will crash in debug build (myapp\_d)  
  
 // // Memory leak: allocate but never free  
 // int \*leak = malloc(sizeof(int));  
 // if (leak != NULL) {  
 // \*leak = 999;  
 // // not freeing on purpose -> leak  
 // }  
  
 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\_multiply(void) {  
 TEST\_ASSERT\_EQUAL(15, multiply(5, 3));  
 TEST\_ASSERT\_EQUAL(-15, multiply(-5, 3));  
 TEST\_ASSERT\_EQUAL(0, multiply(0, 10));  
}  
  
void test\_bad\_add(void) {  
 // This is intentionally wrong, so we just assert what the function actually returns  
 TEST\_ASSERT\_EQUAL(9, bad\_add(5, 3)); // (5 + 3 + 1)  
 TEST\_ASSERT\_EQUAL(-1, bad\_add(-5, 3)); // (-5 + 3 + 1)  
 TEST\_ASSERT\_EQUAL(1, bad\_add(0, 0)); // (0 + 0 + 1)  
}  
  
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);  
 RUN\_TEST(test\_multiply);  
 RUN\_TEST(test\_bad\_add);  
 return UNITY\_END();  
}

## README

# 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.  
  
## Tools and Dependencies  
  
- GNU Make  
- GCC or Clang  
- Address Sanitizer (ASan) for memory error detection  
- gcov and lcov for code coverage  
- gcovr for generating coverage reports  
- pandoc for generating docx reports (optional)  
  
## 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

 ```

## End of Report

Report generated on 09/01/2025 at 20:26:38

## GitHub Info

* GitHub repo name: seanbchaney/makefile-project-starter
* The repository visibility is public.
* The workflow was triggered by seanbchaney

Hash is committed to repo as submission-report-hash.txt

a2e2c166f00a828cbc29e2bbdd734889317a64ef028a52c24f0b9f6ca359f4cf submission-report.md