# Submission Report

* Submission generated at 09/05/2025 at 00:55:47
* Machine info: Linux pkrvm7jw40e0xgp 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/cs452-p0/cs452-p0'  
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/cs452-p0/cs452-p0'  
make BUILD=release  
make[1]: Entering directory '/home/runner/work/cs452-p0/cs452-p0'  
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/cs452-p0/cs452-p0'  
make BUILD=debug-test  
make[1]: Entering directory '/home/runner/work/cs452-p0/cs452-p0'  
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/cs452-p0/cs452-p0'  
make BUILD=test  
make[1]: Entering directory '/home/runner/work/cs452-p0/cs452-p0'  
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/cs452-p0/cs452-p0'  
All builds completed: debug, release, and test.

## Coverage Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:50:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:51:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:52:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:53:test\_multiply:PASS  
  
-----------------------  
4 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:50:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:51:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:52:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:53:test\_multiply:PASS  
  
-----------------------  
4 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 14 14 100%  
------------------------------------------------------------------------------  
TOTAL 14 14 100%  
------------------------------------------------------------------------------

## Address Sanitizer Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:50:test\_get\_greeting:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:51:test\_add:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:52:test\_subtract:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:53:test\_multiply:PASS  
  
-----------------------  
4 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) {  
 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  
  
  
 //greeting--;  
  
 // 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 result.  
 \* @param a The first integer.  
 \* @param b The second integer.  
 \* @return The product of a and b.  
 \*/  
int multiply(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)  
{  
 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, 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);  
 RUN\_TEST(test\_multiply);  
 return UNITY\_END();  
}

## README

# Greeting Message  
## CS452 Project 0  
  
- Name: Benjamin Blodgett  
- Email: benjaminblodgett311@u.boisestate.edu  
- Class: 452-002  
  
## Experience  
  
This project was really easy. I was curious how you could get 100% code coverage so I tried dereferencing a malloc'd pointer in the branch where it returns null. I realized that this block was ommited from testing. I added a bit to this README.md for using live-serever because I think it's more convenient when the coverage files change rather than constantly reopening them in the browser.  
  
## Makefile  
  
To build the project run:  
  
```bash  
make release

To run the executable:

./build/release/myapp

To run the unit tests:

make test && 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

Report watching (run after unit tests):

make report  
cd build/report/html  
live-server . watch=\*.html

```

## End of Report

Report generated on 09/05/2025 at 00:55:48