# Submission Report

* Submission generated at 09/06/2025 at 03:28:47
* 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[1]: Entering directory '/home/runner/work/CS452-Project1/CS452-Project1'  
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-Project1/CS452-Project1'  
make[1]: Entering directory '/home/runner/work/CS452-Project1/CS452-Project1'  
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  
src/lab.c: In function ‘list\_create’:  
src/lab.c:27:28: warning: unused parameter ‘type’ [-Wunused-parameter]  
 27 | List \*list\_create(ListType type) {  
 | ~~~~~~~~~^~~~  
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-Project1/CS452-Project1'  
make[1]: Entering directory '/home/runner/work/CS452-Project1/CS452-Project1'  
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-Project1/CS452-Project1'  
make[1]: Entering directory '/home/runner/work/CS452-Project1/CS452-Project1'  
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-Project1/CS452-Project1'  
Builds completed. You can run the application with: ./build/release/myapp  
You can run the debug build with: ./build/debug/myapp\_d  
You can run the test build with: ./build/tests/myapp\_t  
You can run the debug-test build with: ./build/debug-test/myapp\_td

## Coverage Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:108:test\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:109:test\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:110:test\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:111:test\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:112:test\_list\_edge\_cases:PASS  
  
-----------------------  
5 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:108:test\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:109:test\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:110:test\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:111:test\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:112:test\_list\_edge\_cases: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 63 63 100%  
------------------------------------------------------------------------------  
TOTAL 63 63 100%  
------------------------------------------------------------------------------

## Address Sanitizer Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:108:test\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:109:test\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:110:test\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:111:test\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:112:test\_list\_edge\_cases:PASS  
  
-----------------------  
5 Tests 0 Failures 0 Ignored   
OK

## Source File: lab.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
/\*\*  
 \* Node structure for the circular linked list  
 \* AI Use: Written by own  
 \*/  
typedef struct Node {  
 void \*data;  
 struct Node \*next;  
} Node;  
  
/\*\*  
 \* List structure containing the sentinel node and size  
 \* AI Use: Written by own  
 \*/  
struct List {  
 Node \*head; // Sentinel node  
 size\_t size;  
};  
  
/\*\*  
 \* Creates a new circular list with a sentinel node  
 \* AI Use: Written by own  
 \*/  
List \*list\_create(ListType type) {  
 List \*list = malloc(sizeof(List));  
  
 // Create sentinel node  
 list->head = malloc(sizeof(Node));  
  
 list->head->data = NULL;  
 list->head->next = list->head; // Points to itself when empty  
 list->size = 0;  
  
 return list;  
}  
  
/\*\*  
 \* Destroys the list and frees all memory  
 \* AI Use: Assisted by AI  
 \*/  
void list\_destroy(List \*list, FreeFunc free\_func) {  
 if (!list) return;  
  
 Node \*current = list->head->next;   
 while (current != list->head) {   
 Node \*next = current->next;  
 if (free\_func) {  
 free\_func(current->data);  
 }  
 free(current);  
 current = next;  
 }  
 free(list->head); // Free sentinel node  
 free(list);  
}  
  
/\*\*  
 \* Appends an element to the end of the list  
 \* AI Use: Written by own  
 \*/  
bool list\_append(List \*list, void \*data) {  
 if (!list) return false;  
  
 Node \*new\_node = malloc(sizeof(Node));  
 if (!new\_node) return false;  
  
 new\_node->data = data;  
   
 Node \*current = list->head;  
 while (current->next != list->head) {  
 current = current->next;  
 }  
   
 new\_node->next = list->head; // Point to sentinel  
 current->next = new\_node;  
 list->size++;  
   
 return true;  
}  
  
/\*\*  
 \* Inserts an element at the specified index  
 \* AI Use: Written by own  
 \*/  
bool list\_insert(List \*list, size\_t index, void \*data) {  
 if (!list || index > list->size) return false;  
  
 Node \*new\_node = malloc(sizeof(Node));  
 if (!new\_node) return false;  
  
 new\_node->data = data;  
   
 Node \*current = list->head;  
 for (size\_t i = 0; i < index; i++) {  
 current = current->next;  
 }  
   
 new\_node->next = current->next;  
 current->next = new\_node;  
 list->size++;  
   
 return true;  
}  
  
/\*\*  
 \* Removes and returns the element at the specified index  
 \* AI Use: Assisted by AI  
 \*/  
void \*list\_remove(List \*list, size\_t index) {  
 if (!list || index >= list->size) return NULL;  
  
 Node \*current = list->head;  
 for (size\_t i = 0; i < index; i++) {  
 current = current->next;  
 }  
   
 Node \*to\_remove = current->next;  
 void \*data = to\_remove->data;  
 current->next = to\_remove->next;  
 free(to\_remove);  
 list->size--;  
   
 return data;  
}  
  
/\*\*  
 \* Returns the element at the specified index  
 \* AI Use: Written by own  
 \*/  
void \*list\_get(const List \*list, size\_t index) {  
 if (!list || index >= list->size) return NULL;  
  
 Node \*current = list->head->next; // Skip sentinel  
 for (size\_t i = 0; i < index; i++) {  
 current = current->next;  
 }  
   
 return current->data;  
}  
  
/\*\*  
 \* Returns the current size of the list  
 \* AI Use: Written by own  
 \*/  
size\_t list\_size(const List \*list) {  
 return list ? list->size : 0;  
}  
  
/\*\*  
 \* Checks if the list is empty  
 \* AI Use: Written by own  
 \*/  
bool list\_is\_empty(const List \*list) {  
 return list ? list->size == 0 : true;  
}

## Source File: lab.h

#ifndef LAB\_H  
#define LAB\_H  
  
#include <stdbool.h>  
#include <stddef.h>  
  
/\*\*  
 \* @file lab.h  
 \* @brief Header file for a generic list data structure supporting multiple implementations.  
 \*/  
typedef struct List List;  
  
/\*\*  
 \* @enum ListType  
 \* @brief Enumeration for selecting the list implementation type.  
 \*/  
typedef enum {  
 LIST\_LINKED\_SENTINEL  
} ListType;  
  
/\*\*  
 \* @typedef FreeFunc  
 \* @brief Function pointer type for freeing elements. If NULL, no action is taken.  
 \* Must be provided by the user when destroying the list or removing elements.  
 \*  
 \*/  
typedef void (\*FreeFunc)(void \*);  
  
  
/\*\*  
 \* @brief Create a new list of the specified type.  
 \* @param type The type of list to create (e.g., LIST\_LINKED\_SENTINEL).  
 \* @return Pointer to the newly created list, or NULL on failure.  
 \*/  
List \*list\_create(ListType type);  
  
/\*\*  
 \* @brief Destroy the list and free all associated memory.  
 \* @param list Pointer to the list to destroy.  
 \* @param free\_func Function to free individual elements. If NULL, elements are not freed.  
 \*/  
void list\_destroy(List \*list, FreeFunc free\_func);  
  
/\*\*  
 \* @brief Append an element to the end of the list.  
 \* @param list Pointer to the list.  
 \* @param data Pointer to the data to append.  
 \* @return true on success, false on failure.  
 \*/  
bool list\_append(List \*list, void \*data);  
  
/\*\*  
 \* @brief Insert an element at a specific index.  
 \* @param list Pointer to the list.  
 \* @param index Index at which to insert the element.  
 \* @param data Pointer to the data to insert.  
 \* @return true on success, false on failure (e.g., index out of bounds).  
 \*/  
bool list\_insert(List \*list, size\_t index, void \*data);  
  
/\*\*  
 \* @brief Remove an element at a specific index.  
 \* @param list Pointer to the list.  
 \* @param index Index of the element to remove.  
 \* @return Pointer to the element, or NULL if index is out of bounds.  
 \*/  
void \*list\_remove(List \*list, size\_t index);  
  
/\*\*  
 \* @brief Get a pointer the element at a specific index.  
 \* @param list Pointer to the list.  
 \* @param index Index of the element to retrieve.  
 \* @return Pointer to the element, or NULL if index is out of bounds.  
 \*/  
void \*list\_get(const List \*list, size\_t index);  
  
/\*\*  
 \* @brief Get the current size of the list.  
 \* @param list Pointer to the list.  
 \* @return The number of elements in the list.  
 \*/  
size\_t list\_size(const List \*list);  
  
/\*\*  
 \* @brief Check if the list is empty.  
 \* @param list Pointer to the list.  
 \* @return true if the list is empty, false otherwise.  
 \*/  
bool list\_is\_empty(const List \*list);  
  
#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)  
{  
 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\_list\_create(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list);  
 TEST\_ASSERT\_TRUE(list\_is\_empty(list));  
 TEST\_ASSERT\_EQUAL(0, list\_size(list));  
 list\_destroy(list, NULL);  
}  
  
void test\_list\_append(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 int \*nums[3];  
 for (int i = 0; i < 3; i++) {  
 nums[i] = malloc(sizeof(int));  
 \*nums[i] = i + 1;  
 TEST\_ASSERT\_TRUE(list\_append(list, nums[i]));  
 }  
   
 TEST\_ASSERT\_EQUAL(3, list\_size(list));  
 TEST\_ASSERT\_EQUAL(1, \*(int\*)list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL(2, \*(int\*)list\_get(list, 1));  
 TEST\_ASSERT\_EQUAL(3, \*(int\*)list\_get(list, 2));  
   
 list\_destroy(list, free);  
}  
  
void test\_list\_insert(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
   
 int \*num1 = malloc(sizeof(int));  
 \*num1 = 10;  
 int \*num2 = malloc(sizeof(int));  
 \*num2 = 20;  
   
 TEST\_ASSERT\_TRUE(list\_insert(list, 0, num1));  
 TEST\_ASSERT\_TRUE(list\_insert(list, 1, num2));  
 TEST\_ASSERT\_EQUAL(2, list\_size(list));  
 TEST\_ASSERT\_EQUAL(10, \*(int\*)list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL(20, \*(int\*)list\_get(list, 1));  
   
 int \*num3 = malloc(sizeof(int));  
 \*num3 = 15;  
 TEST\_ASSERT\_TRUE(list\_insert(list, 1, num3));  
 TEST\_ASSERT\_EQUAL(15, \*(int\*)list\_get(list, 1));  
 TEST\_ASSERT\_EQUAL(20, \*(int\*)list\_get(list, 2));  
   
 list\_destroy(list, free);  
}  
  
void test\_list\_remove(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
   
 int \*nums[3];  
 for (int i = 0; i < 3; i++) {  
 nums[i] = malloc(sizeof(int));  
 \*nums[i] = i + 1;  
 list\_append(list, nums[i]);  
 }  
   
 void \*removed = list\_remove(list, 1);  
 TEST\_ASSERT\_NOT\_NULL(removed);  
 TEST\_ASSERT\_EQUAL(2, \*(int\*)removed);  
 free(removed);  
   
 TEST\_ASSERT\_EQUAL(2, list\_size(list));  
 TEST\_ASSERT\_EQUAL(1, \*(int\*)list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL(3, \*(int\*)list\_get(list, 1));  
   
 list\_destroy(list, free);  
}  
  
void test\_list\_edge\_cases(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
   
 // Test NULL data  
 TEST\_ASSERT\_TRUE(list\_append(list, NULL));  
 TEST\_ASSERT\_NULL(list\_get(list, 0));  
   
 // Test invalid index  
 TEST\_ASSERT\_NULL(list\_get(list, 99));  
 TEST\_ASSERT\_FALSE(list\_insert(list, 99, NULL));  
 TEST\_ASSERT\_NULL(list\_remove(list, 99));  
   
 // Test empty list  
 List \*empty\_list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_TRUE(list\_is\_empty(empty\_list));  
 TEST\_ASSERT\_NULL(list\_get(empty\_list, 0));  
 TEST\_ASSERT\_NULL(list\_remove(empty\_list, 0));  
   
 list\_destroy(list, NULL);  
 list\_destroy(empty\_list, NULL);  
}  
  
int main(void) {  
 UNITY\_BEGIN();  
 RUN\_TEST(test\_list\_create);  
 RUN\_TEST(test\_list\_append);  
 RUN\_TEST(test\_list\_insert);  
 RUN\_TEST(test\_list\_remove);  
 RUN\_TEST(test\_list\_edge\_cases);  
 return UNITY\_END();  
}

## README

# Project 1 - Simple Linked List  
  
- Name: Anirud Shrestha  
- Email: anirudshrestha@u.boisestate.edu  
- Class: 452-002  
  
## Known Bugs or Issues  
  
There not known bugs or issues.  
  
## Experience  
  
This project was not too bad given that I had good understanding of what single linked list was and how it should work. I did have spend time google and asking AI on how pointer work like deferencing and free-ing up the memory of pointers. I also had to lookup about "sentinel" node which was something I hadn't heard off. This is cool approach. Over all, it was a good project. I didn't have any leaks or issues.

## End of Report

Report generated on 09/06/2025 at 03:28:48

## GitHub Info

* GitHub repo name: Anirud2002/CS452-Project1
* The repository visibility is public.
* The workflow was triggered by Anirud2002

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

744ef79b1b2986886da6a41103b8064796e5977876bca285fedc3fb26e31be08 submission-report.md 10abf8962cdef1eb56ee551d56d75b8772cacbb9374015342f6274ea3d2c7384 submission-report.md