# Submission Report

* Submission generated at 09/08/2025 at 05:36:47
* 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/makefile-project/makefile-project'  
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/makefile-project'  
make[1]: Entering directory '/home/runner/work/makefile-project/makefile-project'  
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/makefile-project'  
make[1]: Entering directory '/home/runner/work/makefile-project/makefile-project'  
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/makefile-project'  
make[1]: Entering directory '/home/runner/work/makefile-project/makefile-project'  
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/makefile-project'  
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

tests/lab-test.c:127:test\_create\_and\_destroy:PASS  
tests/lab-test.c:128:test\_append\_and\_get:PASS  
tests/lab-test.c:129:test\_insert:PASS  
tests/lab-test.c:130:test\_remove:PASS  
tests/lab-test.c:131:test\_invalid\_ops:PASS  
tests/lab-test.c:132:test\_destroy\_with\_free\_func:PASS  
  
-----------------------  
6 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
tests/lab-test.c:127:test\_create\_and\_destroy:PASS  
tests/lab-test.c:128:test\_append\_and\_get:PASS  
tests/lab-test.c:129:test\_insert:PASS  
tests/lab-test.c:130:test\_remove:PASS  
tests/lab-test.c:131:test\_invalid\_ops:PASS  
tests/lab-test.c:132:test\_destroy\_with\_free\_func:PASS  
  
-----------------------  
6 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 72 70 97% 38-39  
------------------------------------------------------------------------------  
TOTAL 72 70 97%  
------------------------------------------------------------------------------

## Address Sanitizer Report

tests/lab-test.c:127:test\_create\_and\_destroy:PASS  
tests/lab-test.c:128:test\_append\_and\_get:PASS  
tests/lab-test.c:129:test\_insert:PASS  
tests/lab-test.c:130:test\_remove:PASS  
tests/lab-test.c:131:test\_invalid\_ops:PASS  
tests/lab-test.c:132:test\_destroy\_with\_free\_func:PASS  
  
-----------------------  
6 Tests 0 Failures 0 Ignored   
OK

## Source File: lab.c

#include "lab.h"  
#include <stdlib.h>  
#include <stdio.h>  
  
/\*\*  
 \* @file lab.c  
 \* @brief Implementation of a circular doubly linked list with a sentinel node.  
 \*/  
  
/\*   
 \* Internal node structure for the list  
 \*/  
typedef struct Node {  
 void \*data;  
 struct Node \*next;  
 struct Node \*prev;  
} Node;  
  
/\*   
 \* The List structure definition (hidden from lab.h).  
 \*/  
struct List {  
 ListType type;  
 size\_t size;  
 Node \*sentinel;  
};  
  
/\*\*  
 \* Create a new list of the specified type.  
 \* AI Use: Written By AI  
 \*/  
List \*list\_create(ListType type) {  
 List \*list = malloc(sizeof(List));  
 if (!list) return NULL;  
  
 Node \*sentinel = malloc(sizeof(Node));  
 if (!sentinel) {  
 free(list);  
 return NULL;  
 }  
  
 sentinel->data = NULL;  
 sentinel->next = sentinel;  
 sentinel->prev = sentinel;  
  
 list->type = type;  
 list->size = 0;  
 list->sentinel = sentinel;  
 return list;  
}  
  
/\*\*  
 \* Destroy the list and free all associated memory.  
 \* AI Use: Written By AI  
 \*/  
void list\_destroy(List \*list, FreeFunc free\_func) {  
 if (!list) return;  
  
 Node \*cur = list->sentinel->next;  
 while (cur != list->sentinel) {  
 Node \*next = cur->next;  
 if (free\_func) free\_func(cur->data);  
 free(cur);  
 cur = next;  
 }  
  
 free(list->sentinel);  
 free(list);  
}  
  
/\*\*  
 \* Append an element to the end of the list.  
 \* AI Use: Written By AI  
 \*/  
bool list\_append(List \*list, void \*data) {  
 if (!list) return false;  
  
 Node \*node = malloc(sizeof(Node));  
 if (!node) return false;  
 node->data = data;  
  
 Node \*tail = list->sentinel->prev;  
  
 tail->next = node;  
 node->prev = tail;  
 node->next = list->sentinel;  
 list->sentinel->prev = node;  
  
 list->size++;  
 return true;  
}  
  
/\*\*  
 \* Insert an element at a specific index.  
 \* AI Use: Written By AI  
 \*/  
bool list\_insert(List \*list, size\_t index, void \*data) {  
 if (!list || index > list->size) return false;  
  
 Node \*node = malloc(sizeof(Node));  
 if (!node) return false;  
 node->data = data;  
  
 Node \*cur = list->sentinel;  
 for (size\_t i = 0; i < index; i++) {  
 cur = cur->next;  
 }  
  
 Node \*next = cur->next;  
 cur->next = node;  
 node->prev = cur;  
 node->next = next;  
 next->prev = node;  
  
 list->size++;  
 return true;  
}  
  
/\*\*  
 \* Remove an element at a specific index.  
 \* AI Use: Written By AI  
 \*/  
void \*list\_remove(List \*list, size\_t index) {  
 if (!list || index >= list->size) return NULL;  
  
 Node \*cur = list->sentinel->next;  
 for (size\_t i = 0; i < index; i++) {  
 cur = cur->next;  
 }  
  
 void \*data = cur->data;  
 cur->prev->next = cur->next;  
 cur->next->prev = cur->prev;  
 free(cur);  
  
 list->size--;  
 return data;  
}  
  
/\*\*  
 \* Get a pointer to the element at a specific index.  
 \* AI Use: Written By AI  
 \*/  
void \*list\_get(const List \*list, size\_t index) {  
 if (!list || index >= list->size) return NULL;  
  
 Node \*cur = list->sentinel->next;  
 for (size\_t i = 0; i < index; i++) {  
 cur = cur->next;  
 }  
 return cur->data;  
}  
  
/\*\*  
 \* Get the current size of the list.  
 \* AI Use: Written By AI  
 \*/  
size\_t list\_size(const List \*list) {  
 return list ? list->size : 0;  
}  
  
/\*\*  
 \* Check if the list is empty.  
 \* AI Use: Written By AI  
 \*/  
bool list\_is\_empty(const List \*list) {  
 return !list || list->size == 0;  
}

## 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)  
{  
 printf("Hello World");  
 return 0;  
}

## Test Files

### lab-test.c

#include <stdlib.h>  
#include <stdio.h>  
#include "harness/unity.h"  
#include "../src/lab.h"  
  
void setUp(void) {  
}  
  
void tearDown(void) {  
}  
  
void test\_create\_and\_destroy(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list);  
 TEST\_ASSERT\_TRUE(list\_is\_empty(list));  
 TEST\_ASSERT\_EQUAL\_UINT32(0, list\_size(list));  
 list\_destroy(list, NULL);  
}  
  
void test\_append\_and\_get(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 int a = 10, b = 20, c = 30;  
  
 TEST\_ASSERT\_TRUE(list\_append(list, &a));  
 TEST\_ASSERT\_TRUE(list\_append(list, &b));  
 TEST\_ASSERT\_TRUE(list\_append(list, &c));  
  
 TEST\_ASSERT\_EQUAL\_UINT32(3, list\_size(list));  
 TEST\_ASSERT\_EQUAL\_PTR(&a, list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL\_PTR(&b, list\_get(list, 1));  
 TEST\_ASSERT\_EQUAL\_PTR(&c, list\_get(list, 2));  
  
 list\_destroy(list, NULL);  
}  
  
void test\_insert(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 int a = 1, b = 2, c = 3, d = 4;  
  
 list\_append(list, &a); // [1]  
 list\_append(list, &c); // [1,3]  
 TEST\_ASSERT\_TRUE(list\_insert(list, 1, &b)); // [1,2,3]  
 TEST\_ASSERT\_TRUE(list\_insert(list, 3, &d)); // [1,2,3,4]  
  
 TEST\_ASSERT\_EQUAL\_UINT32(4, list\_size(list));  
 TEST\_ASSERT\_EQUAL\_PTR(&a, list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL\_PTR(&b, list\_get(list, 1));  
 TEST\_ASSERT\_EQUAL\_PTR(&c, list\_get(list, 2));  
 TEST\_ASSERT\_EQUAL\_PTR(&d, list\_get(list, 3));  
  
 list\_destroy(list, NULL);  
}  
  
void test\_remove(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 int a = 5, b = 6, c = 7;  
  
 list\_append(list, &a);  
 list\_append(list, &b);  
 list\_append(list, &c);  
  
 void \*removed = list\_remove(list, 1); // remove middle  
 TEST\_ASSERT\_EQUAL\_PTR(&b, removed);  
 TEST\_ASSERT\_EQUAL\_UINT32(2, list\_size(list));  
 TEST\_ASSERT\_EQUAL\_PTR(&a, list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL\_PTR(&c, list\_get(list, 1));  
  
 removed = list\_remove(list, 0); // remove head  
 TEST\_ASSERT\_EQUAL\_PTR(&a, removed);  
 TEST\_ASSERT\_EQUAL\_UINT32(1, list\_size(list));  
  
 removed = list\_remove(list, 0); // remove last  
 TEST\_ASSERT\_EQUAL\_PTR(&c, removed);  
 TEST\_ASSERT\_TRUE(list\_is\_empty(list));  
  
 list\_destroy(list, NULL);  
}  
  
void test\_invalid\_ops(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
  
 // Invalid get  
 TEST\_ASSERT\_NULL(list\_get(list, 0));  
  
 // Invalid remove  
 TEST\_ASSERT\_NULL(list\_remove(list, 0));  
  
 // Invalid insert (too high index)  
 int x = 42;  
 TEST\_ASSERT\_FALSE(list\_insert(list, 5, &x));  
  
 list\_destroy(list, NULL);  
}  
  
static int free\_count = 0;  
  
void custom\_free(void \*ptr) {  
 if (ptr) {  
 free(ptr);  
 free\_count++;  
 }  
}  
  
void test\_destroy\_with\_free\_func(void) {  
 free\_count = 0;  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
  
 int \*a = malloc(sizeof(int));  
 int \*b = malloc(sizeof(int));  
 int \*c = malloc(sizeof(int));  
  
 \*a = 10; \*b = 20; \*c = 30;  
  
 list\_append(list, a);  
 list\_append(list, b);  
 list\_append(list, c);  
  
 TEST\_ASSERT\_EQUAL\_UINT32(3, list\_size(list));  
  
 list\_destroy(list, custom\_free);  
  
 TEST\_ASSERT\_EQUAL\_INT(3, free\_count); // all freed  
}  
  
int main(void) {  
 UNITY\_BEGIN();  
 RUN\_TEST(test\_create\_and\_destroy);  
 RUN\_TEST(test\_append\_and\_get);  
 RUN\_TEST(test\_insert);  
 RUN\_TEST(test\_remove);  
 RUN\_TEST(test\_invalid\_ops);  
 RUN\_TEST(test\_destroy\_with\_free\_func);  
 return UNITY\_END();  
}```  
  
  
  
---  
  
## README  
```markdown  
# Project 1 – Simple Linked List  
  
## Overview  
This project implements a \*\*circular doubly linked list with a sentinel node\*\* in C.   
It is the first project in the course and serves as a warm-up to practice:  
- Pointers and dynamic memory management  
- Structs and modular design in C  
- Writing portable code that compiles in both \*\*GitHub Codespaces\*\* and \*\*Onyx\*\*  
- Unit testing with the Unity framework  
  
  
---  
  
## Features  
- Circular doubly linked list with a sentinel node  
- Generic storage (`void \*` for any data type)  
- Append, insert, remove, and get functions  
- Query functions: size, is\_empty  
- Optional `FreeFunc` to safely free stored elements  
- Thorough unit testing with Unity  
  
---  
  
## Build and Run  
  
From the root of the repository:  
  
### Build everything  
```bash  
make all

## End of Report

Report generated on 09/08/2025 at 05:36:49

## GitHub Info

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

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

1c2875a878ef5ab29fb2158e48705e9a00316dae227bf18d77b53abbbe55a578 submission-report.md