# Submission Report

* Submission generated at 09/05/2025 at 16:21:25
* 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

No main.c found in src. Skipping debug and release builds.  
make[1]: Entering directory '/home/runner/work/OS-P1/OS-P1'  
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/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/OS-P1/OS-P1'  
make[1]: Entering directory '/home/runner/work/OS-P1/OS-P1'  
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/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/OS-P1/OS-P1'  
Test builds completed. 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:124:test\_list\_basic:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:125:test\_list\_null\_and\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:126:test\_list\_insert\_remove\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:127:test\_list\_destroy\_free\_func:PASS  
  
-----------------------  
4 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:124:test\_list\_basic:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:125:test\_list\_null\_and\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:126:test\_list\_insert\_remove\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:127:test\_list\_destroy\_free\_func: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 80 80 100%  
------------------------------------------------------------------------------  
TOTAL 80 80 100%  
------------------------------------------------------------------------------

## Address Sanitizer Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:124:test\_list\_basic:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:125:test\_list\_null\_and\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:126:test\_list\_insert\_remove\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:127:test\_list\_destroy\_free\_func:PASS  
  
-----------------------  
4 Tests 0 Failures 0 Ignored   
OK

## Source File: lab.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
/\*\*  
 \* Constructs the node struct.  
 \* AI use: Assisted by AI  
 \*/  
typedef struct Node{  
 void \*data;  
 struct Node \*next;  
 struct Node \*prev;  
}Node;  
  
/\*\*  
 \* Constructs the list struct  
 \* AI use: No AI  
 \*/  
struct List {  
 Node \*SENTINEL;  
 size\_t size;  
 ListType type;  
};  
  
/\*\*  
 \* Creates a list and sets the Sentinel node.  
 \* AI usage: written by AI  
 \*/  
List \*list\_create(ListType type) {  
 List \*list = malloc(sizeof(List));  
 Node \*sentinel = malloc(sizeof(Node));  
  
 sentinel->next = sentinel;  
 sentinel->prev = sentinel;  
 list->SENTINEL = sentinel;  
 list->type = type;  
 list->size = 0;  
 return list;  
}  
  
/\*\*  
 \* Destroys a list and sets everything free.  
 \* AI use: Assisted by AI  
 \*/  
void list\_destroy(List \*list, FreeFunc free\_func) {  
 Node \*first = list->SENTINEL->next;  
 while(first != list->SENTINEL){  
 Node \*next = first->next;  
 if(free\_func != NULL && first->data != NULL){  
 free\_func(first->data);  
 free(first);  
 }  
 first = next;  
 }  
 free(list->SENTINEL);  
 free(list);  
}  
  
/\*\*  
 \* Adds a new node to the end of a list.  
 \* Returns true if successful and updates the length of the list.  
 \* AI use: Assisted by AI  
 \*/  
bool list\_append(List \*list, void \*data){  
 if(list == NULL){  
 return false;  
 }  
 Node \*newNode = malloc(sizeof(Node));  
 newNode->data = data;  
  
 Node \*currentLast = list->SENTINEL->prev;  
   
 newNode->prev = currentLast;  
 currentLast->next = newNode;  
 newNode->next = list->SENTINEL;  
 list->SENTINEL->prev = newNode;  
 list->size++;  
 return true;  
}  
  
/\*\*  
 \* Inserts a node into a list at a specific index.   
 \* Returns true if the index exists.  
 \* AI use: No AI  
 \*/  
bool list\_insert(List \*list, size\_t index, void \*data){  
 if(list == NULL){  
 return false;  
 }  
 Node \*newNode = malloc(sizeof(Node));  
 newNode->data = data;  
  
 Node \*current = list->SENTINEL->next;  
  
 if(index >= list->size){  
 free(newNode);  
 return false;  
 }  
 for(size\_t i = 0; i<index; i++){  
 current = current->next;  
 }  
   
 newNode->next = current;  
 newNode->prev = current->prev;  
 current->prev->next = newNode;  
 current->prev = newNode;  
 list->size++;  
 return true;  
}  
  
/\*\*  
 \* Removes a specific node in a list based on index.  
 \* AI use: Assisted by AI  
 \*/  
void \*list\_remove(List \*list, size\_t index){  
 if(list == NULL){  
 return false;  
 }  
 if(index >= list->size){  
 return NULL;  
 }  
  
 Node \*current = list->SENTINEL->next;  
 for(size\_t i = 0; i<index; i++){  
 current = current->next;  
 }  
   
 current->next->prev = current->prev;  
 current->prev->next = current->next;  
 void \*data = current->data;  
 free(current);  
 list->size--;  
 return data;  
}  
  
/\*\*  
 \* Gets a specific node in the list based on index.  
 \* AI use: Assisted by AI  
 \*/  
void \*list\_get(const List \*list, size\_t index){  
 if(list == NULL){  
 return NULL;  
 }  
 if(index >= list->size){  
 return NULL;  
 }  
  
 Node \*current = list->SENTINEL->next;  
  
 for(size\_t i = 0; i<index; i++){  
 current = current->next;  
 }  
 // Node \*data = current->data;  
 return current->data;  
}  
  
/\*\*  
 \* Returns the size of the list if not null.  
 \* AI use: No AI  
 \*/  
size\_t list\_size(const List \*list){  
 if(list == NULL){  
 return false;  
 }  
  
 return list->size;  
}  
  
/\*\*  
 \* Checks to see if the list is empty and returns true if it is.  
 \* AI use: Assisted by AI  
 \*/  
bool list\_is\_empty(const List \*list){  
 if(list == NULL){  
 return false;  
 }  
 return 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

## 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\_basic(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));  
  
 int \*val1 = malloc(sizeof(int));  
 int \*val2 = malloc(sizeof(int));  
 \*val1 = 42;  
 \*val2 = 99;  
  
 TEST\_ASSERT\_TRUE(list\_append(list, val1));  
 TEST\_ASSERT\_FALSE(list\_is\_empty(list));  
 TEST\_ASSERT\_EQUAL(1, list\_size(list));  
 TEST\_ASSERT\_EQUAL\_PTR(val1, list\_get(list, 0));  
  
 TEST\_ASSERT\_TRUE(list\_insert(list, 0, val2));  
 TEST\_ASSERT\_EQUAL(2, list\_size(list));  
 TEST\_ASSERT\_EQUAL\_PTR(val2, list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL\_PTR(val1, list\_get(list, 1));  
  
 int \*removed = list\_remove(list, 0);  
 TEST\_ASSERT\_EQUAL\_PTR(val2, removed);  
 free(removed);  
 TEST\_ASSERT\_EQUAL(1, list\_size(list));  
  
 list\_destroy(list, free);  
}  
  
  
void test\_list\_null\_and\_bounds(void) {  
 // Test NULL list handling  
 TEST\_ASSERT\_FALSE(list\_append(NULL, NULL));  
 TEST\_ASSERT\_FALSE(list\_insert(NULL, 0, NULL));  
 TEST\_ASSERT\_NULL(list\_remove(NULL, 0));  
 TEST\_ASSERT\_NULL(list\_get(NULL, 0));  
 TEST\_ASSERT\_EQUAL(0, list\_size(NULL));  
 TEST\_ASSERT\_FALSE(list\_is\_empty(NULL));  
  
 // Create a list and test out-of-bounds  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list);  
  
 int \*val = malloc(sizeof(int));  
 \*val = 123;  
 TEST\_ASSERT\_TRUE(list\_append(list, val));  
 TEST\_ASSERT\_NULL(list\_get(list, 1));  
 TEST\_ASSERT\_FALSE(list\_insert(list, 2, val));  
 TEST\_ASSERT\_NULL(list\_remove(list, 2));  
  
 list\_destroy(list, free);  
}  
  
void test\_list\_insert\_remove\_get(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list);  
  
 int \*a = malloc(sizeof(int));  
 int \*b = malloc(sizeof(int));  
 int \*c = malloc(sizeof(int));  
 \*a = 1; \*b = 2; \*c = 3;  
  
 // Insert at index 0 (should fail, list is empty)  
 TEST\_ASSERT\_FALSE(list\_insert(list, 1, a));  
 TEST\_ASSERT\_TRUE(list\_append(list, a));  
 TEST\_ASSERT\_TRUE(list\_insert(list, 0, b)); // Insert at front  
 TEST\_ASSERT\_TRUE(list\_insert(list, 1, c)); // Insert in middle  
  
 TEST\_ASSERT\_EQUAL\_PTR(b, list\_get(list, 0));  
 TEST\_ASSERT\_EQUAL\_PTR(c, list\_get(list, 1));  
 TEST\_ASSERT\_EQUAL\_PTR(a, list\_get(list, 2));  
  
 // Remove from middle  
 int \*removed = list\_remove(list, 1);  
 TEST\_ASSERT\_EQUAL\_PTR(c, removed);  
 free(removed);  
  
 // Remove from front  
 removed = list\_remove(list, 0);  
 TEST\_ASSERT\_EQUAL\_PTR(b, removed);  
 free(removed);  
  
 // Remove last  
 removed = list\_remove(list, 0);  
 TEST\_ASSERT\_EQUAL\_PTR(a, removed);  
 free(removed);  
  
 TEST\_ASSERT\_TRUE(list\_is\_empty(list));  
 list\_destroy(list, NULL);  
}  
  
void test\_list\_destroy\_free\_func(void) {  
 List \*list = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list);  
  
 int \*x = malloc(sizeof(int));  
 int \*y = malloc(sizeof(int));  
 \*x = 10; \*y = 20;  
  
 TEST\_ASSERT\_TRUE(list\_append(list, x));  
 TEST\_ASSERT\_TRUE(list\_append(list, y));  
  
 // Destroy with free\_func  
 list\_destroy(list, free);  
}  
  
  
int main(void) {  
 UNITY\_BEGIN();  
 RUN\_TEST(test\_list\_basic);  
 RUN\_TEST(test\_list\_null\_and\_bounds);  
 RUN\_TEST(test\_list\_insert\_remove\_get);  
 RUN\_TEST(test\_list\_destroy\_free\_func);  
 return UNITY\_END();  
}

## README

Name: Brooke Matthews  
Email: brookematthews@u.boisestate.edu  
Class: 452-002  
  
Known Bugs or Issues  
There are no known bugs or issues. I checked and I was passing all of my tests and the report showed 100% code coverage with no issues.  
  
Experience  
This was a fun asssignement. I was worried at first since it had been a little while since I had programmed in C, but it ended up being fine! I found that the main questions I ended up asking chat where just reminders about how to set up structs and syntactical questions. Once I completed a couple functions I found my rhythm and was able to complete most of them with little trouble. I did have to back and debug some of my functions beacause I was adding a lot of null checks where I didnt need them. For the most part this was a good project in remembering C.

## End of Report

Report generated on 09/05/2025 at 16:21:26

## GitHub Info

* GitHub repo name: bjBSU/OS-P1
* The repository visibility is public.
* The workflow was triggered by bjBSU

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

d48c893ede72219f07111fbd487545cbc4e8454555b93eb586e2da602a8f9e1d submission-report.md