# Submission Report

* Submission generated at 09/08/2025 at 05:23:31
* 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/P1-Simple-Linked-List/P1-Simple-Linked-List'  
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/P1-Simple-Linked-List/P1-Simple-Linked-List'  
make[1]: Entering directory '/home/runner/work/P1-Simple-Linked-List/P1-Simple-Linked-List'  
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/P1-Simple-Linked-List/P1-Simple-Linked-List'  
make[1]: Entering directory '/home/runner/work/P1-Simple-Linked-List/P1-Simple-Linked-List'  
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/P1-Simple-Linked-List/P1-Simple-Linked-List'  
make[1]: Entering directory '/home/runner/work/P1-Simple-Linked-List/P1-Simple-Linked-List'  
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/P1-Simple-Linked-List/P1-Simple-Linked-List'  
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:163:test\_func\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:164:test\_func\_list\_destroy:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:165:test\_func\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:166:test\_func\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:167:test\_func\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:168:test\_func\_list\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:169:test\_func\_list\_is\_empty:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:170:test\_func\_list\_insert\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:171:test\_func\_list\_remove\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:172:test\_func\_list\_get\_out\_of\_bounds:PASS  
  
-----------------------  
10 Tests 0 Failures 0 Ignored   
OK  
./build/tests/myapp\_t  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:163:test\_func\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:164:test\_func\_list\_destroy:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:165:test\_func\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:166:test\_func\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:167:test\_func\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:168:test\_func\_list\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:169:test\_func\_list\_is\_empty:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:170:test\_func\_list\_insert\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:171:test\_func\_list\_remove\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:172:test\_func\_list\_get\_out\_of\_bounds:PASS  
  
-----------------------  
10 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 86 80 93% 26,33-34,46,77,106  
------------------------------------------------------------------------------  
TOTAL 86 80 93%  
------------------------------------------------------------------------------

## Address Sanitizer Report

Setting up tests...  
Tearing down tests...  
tests/lab-test.c:163:test\_func\_list\_create:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:164:test\_func\_list\_destroy:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:165:test\_func\_list\_append:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:166:test\_func\_list\_insert:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:167:test\_func\_list\_remove:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:168:test\_func\_list\_get:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:169:test\_func\_list\_is\_empty:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:170:test\_func\_list\_insert\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:171:test\_func\_list\_remove\_out\_of\_bounds:PASS  
Setting up tests...  
Tearing down tests...  
tests/lab-test.c:172:test\_func\_list\_get\_out\_of\_bounds:PASS  
  
-----------------------  
10 Tests 0 Failures 0 Ignored   
OK

## Source File: lab.c

#include "lab.h"  
#include <stdio.h>  
#include <stdlib.h>  
  
typedef struct Node {  
 void \*data;  
 struct Node \*prev;  
 struct Node \*next;  
} Node;  
  
typedef struct List {  
 Node \*sentinel;  
 Node \*head;  
 Node \*tail;   
 size\_t size;  
} List;  
  
/\*\*  
\* Creates a List Object  
\* AI Use: No AI  
\*/  
List \*list\_create(ListType type) {  
 // Create List Object  
 List \*list = malloc(sizeof(List));  
 if (!list) {  
 return NULL;  
 }  
  
 if(type == LIST\_LINKED\_SENTINEL) {  
 // Create senitnel Node  
 Node \*sentinelNode = malloc(sizeof(Node));  
 if (!sentinelNode) {  
 free(list);  
 return NULL;  
 }  
  
 // Setup sentinel Node  
 sentinelNode->data = NULL;  
 sentinelNode->prev = sentinelNode;  
 sentinelNode->next = sentinelNode;  
 list->sentinel = sentinelNode;  
 list->size = 0;  
  
 return list;  
 } else {  
 return NULL;  
 }  
}  
  
/\*\*  
\* Destroys a List Object  
\* AI Use: Assisted by AI  
\*/  
void list\_destroy(List \*list, FreeFunc free\_func) {  
 Node \*current = list->sentinel->next;  
 while (current != list->sentinel) {  
 Node \*nextNode = current->next;  
 if (free\_func) {  
 int \*ptr\_1 = current->data;  
 free\_func(ptr\_1);  
 }  
 free(current);  
 current = nextNode;  
 }  
 free(list->sentinel);  
 free(list);  
}  
  
/\*\*  
\* Appends to the end of the List  
\* AI Use: Assisted by AI  
\*/  
bool list\_append(List \*list, void \*data) {  
 // Initialize new node  
 Node \*newNode = malloc(sizeof(Node));  
 if (!newNode) {  
 return false;  
 }  
 newNode->data = data;  
  
 // Append new node   
 Node \*sentinel = list->sentinel;  
 Node \*tailNode = sentinel->prev;  
 newNode->next = sentinel;  
 newNode->prev = tailNode;  
 tailNode->next = newNode;  
 sentinel->prev = newNode;  
  
 list->size++;  
 return true;  
}  
  
/\*\*  
\* Inserts an Object into a List  
\* AI Use: Assisted by AI  
\*/  
bool list\_insert(List \*list, size\_t index, void \*data) {  
 // Check out-of-bounds errors  
 if (index > list->size) {  
 return false;  
 }  
   
 // Initialize node  
 Node \*newNode = malloc(sizeof(Node));  
 if (!newNode) {  
 return false;  
 }  
  
 newNode->data = data;  
  
 // Find index  
 Node \*current = list->sentinel->next;  
 for (size\_t i = 0; i < index; i++) {  
 current = current->next;  
 }  
  
 // Insert node before 'current'  
 newNode->prev = current->prev;  
 newNode->next = current;  
 current->prev->next = newNode;  
 current->prev = newNode;  
  
 list->size++;  
 return 0;  
}  
  
/\*\*  
\* Removes an Object from a List  
\* AI Use: No AI  
\*/  
void \*list\_remove(List \*list, size\_t index) {  
 // Check if index is out-of-bounds  
 if (index >= list->size) {  
 return NULL;  
 }  
  
 // Traverse to Node by index  
 Node \*current = list->sentinel->next;  
 for (size\_t i = 0; i < index; i++) {  
 current = current->next;  
 }  
  
 // Remove Node at the index  
 current->prev->next = current->next;  
 current->next->prev = current->prev;  
 void \*data = current->data;  
 free(current);  
  
 list->size--;  
 return data; // This is a pointer to the removed element!  
}  
  
  
/\*\*  
\* Gets the Object at an index of a List  
\* AI Use: Assisted by AI  
\*/  
void \*list\_get(const List \*list, size\_t index) {  
 // Check index out-of-bounds errors  
 if (index >= list->size) {  
 return NULL;  
 }  
   
 // Determine fastest direction to index (forwards/backwards)  
 Node \*currentNode; size\_t i;  
   
 if (index < list->size / 2) {  
 currentNode = list->sentinel->next;  
 for (i = 0; i < index; i++) {  
 currentNode = currentNode->next;  
 }  
 } else {  
 currentNode = list->sentinel->prev;  
 for (i = list->size - 1; i > index; i--) {  
 currentNode = currentNode->prev;  
 }  
 }  
  
 return currentNode->data;  
}  
  
/\*\*  
 \* Gets the size of a List  
 \* AI Use: No AI  
 \*/  
size\_t list\_size(const List \*list) {  
 return list->size;  
}  
  
/\*\*  
 \* Gets the size of a List  
 \* AI Use: No AI  
 \*/  
bool list\_is\_empty(const List \*list) {  
 if (list->size==0) {  
 return true;  
 }   
 return false;  
}

## 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\_func\_list\_create(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_NOT\_NULL(list\_1);  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_destroy(void) {  
 char \*myString = (char \*)malloc(sizeof(char) \* (15));  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1,myString);  
 FreeFunc func\_1 = free;  
 list\_destroy(list\_1, free); // Mainly test for bugs/crashes  
 list\_1=NULL;  
 TEST\_ASSERT\_NULL(list\_1);  
}  
  
void test\_func\_list\_append(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 TEST\_ASSERT\_EQUAL\_INT(3, list\_size(list\_1));  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_insert(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 int num\_4 = 11;  
 int num\_5 = 29;  
 int num\_6 = 29;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 list\_insert(list\_1, 0, &num\_5);  
 list\_insert(list\_1, 3, &num\_6);  
 TEST\_ASSERT\_EQUAL\_INT(6, list\_size(list\_1));  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_remove(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 int num\_4 = 11;  
 int num\_5 = 23;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 list\_append(list\_1, &num\_5);  
 list\_remove(list\_1, 1);  
 list\_remove(list\_1, 3);  
 TEST\_ASSERT\_EQUAL\_INT(3, list\_size(list\_1));  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_get(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 21;  
 int num\_4 = 11;  
 int num\_5 = 45;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 list\_append(list\_1, &num\_5);   
 int \*value\_1 = list\_get(list\_1, 1);  
 int \*value\_2 = list\_get(list\_1, 3);  
 TEST\_ASSERT\_EQUAL\_INT(6, \*value\_1);  
 TEST\_ASSERT\_EQUAL\_INT(11, \*value\_2);  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_is\_empty(void) {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 21;  
 int num\_4 = 11;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 TEST\_ASSERT\_TRUE(list\_is\_empty(list\_1));  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 TEST\_ASSERT\_FALSE(list\_is\_empty(list\_1));  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_insert\_out\_of\_bounds() {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 int num\_4 = 11;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 bool isSuccess = list\_insert(list\_1, 11, &num\_4);  
 TEST\_ASSERT\_FALSE(isSuccess);  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_remove\_out\_of\_bounds() {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 int num\_4 = 11;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 bool isSuccess = list\_remove(list\_1, 11);  
 TEST\_ASSERT\_FALSE(isSuccess);  
 list\_destroy(list\_1, NULL);  
}  
  
void test\_func\_list\_get\_out\_of\_bounds() {  
 int num\_1 = 5;  
 int num\_2 = 6;  
 int num\_3 = 4;  
 int num\_4 = 11;  
 List \*list\_1 = list\_create(LIST\_LINKED\_SENTINEL);  
 list\_append(list\_1, &num\_1);  
 list\_append(list\_1, &num\_2);  
 list\_append(list\_1, &num\_3);  
 list\_append(list\_1, &num\_4);  
 bool isSuccess = list\_get(list\_1, 22);  
 TEST\_ASSERT\_FALSE(isSuccess);  
 list\_destroy(list\_1, NULL);  
}  
  
int main(void) {  
 UNITY\_BEGIN();  
 RUN\_TEST(test\_func\_list\_create);  
 RUN\_TEST(test\_func\_list\_destroy);  
 RUN\_TEST(test\_func\_list\_append);  
 RUN\_TEST(test\_func\_list\_insert);  
 RUN\_TEST(test\_func\_list\_remove);  
 RUN\_TEST(test\_func\_list\_get);  
 RUN\_TEST(test\_func\_list\_is\_empty);  
 RUN\_TEST(test\_func\_list\_insert\_out\_of\_bounds);  
 RUN\_TEST(test\_func\_list\_remove\_out\_of\_bounds);  
 RUN\_TEST(test\_func\_list\_get\_out\_of\_bounds);  
 return UNITY\_END();  
}

## README

# Project 1 - Simple Linked List  
  
- Name: Martin Guzman  
- Email: martinguzman@u.boisestate.edu  
- Class: 452-002  
  
## Known Bugs or Issues  
  
In this project I had a few small issues. However, they were mainly related to the nuances of coding in the C programming language and were easy to solve with a quick Google search. One issue that I really struggled with was passing pointers through functions as arguments but I eventually succeded in implementing those fuunctions.  
  
## Experience  
  
Overall I really enjoyed this project. Although I did not learn too much about operating systems or any material that was the focus of this class, I found this exercise to be a confidence booster with the language C as we continue to use it in this class. I also feel much more comfortable using the tools such as the Unity framework and Codespace.

## End of Report

Report generated on 09/08/2025 at 05:23:32

## GitHub Info

* GitHub repo name: BlueBronco06/P1-Simple-Linked-List
* The repository visibility is public.
* The workflow was triggered by BlueBronco06

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

9e491244bf06b4ce4645ad4f5b28df9b552606a5ce18b29637655d9b999e4b8c submission-report.md