Exercise 4: Memory Management with Valgrind

Detecting and Fixing Memory Leaks

ZYAD FRI
Computer Science - UM6P

January 29, 2026

1 Problem Analysis

1.1 Original Code Issues

Issue 1: Empty free_memory() function

```
void free_memory(int *arr) {
    // Empty - memory not freed!
}
```

Issue 2: Missing free for array_copy

- `array_copy` allocated in `duplicate_array()`
- Never freed before program exit

1.2 Memory Allocation Summary

Table 1: Memory Allocations in Original Program

Allocation	Location	Size	Status
array	line 48	20 bytes	Not freed
array_copy	line 51	20 bytes	Not freed
printf buffer	internal	1024 bytes	Freed
Total		**1064 bytes**	**40 bytes leaked**

2 Valgrind Analysis

2.1 Compilation and Execution

```
gcc -g -o memory_debug memory_debug.c
valgrind --leak-check=full --track-origins=yes ./memory_debug
```

2.2 Valgrind Output - Before Fix

```
==31110== HEAP SUMMARY:
==31110==     in use at exit: 40 bytes in 2 blocks
==31110==   total heap usage: 3 allocs, 1 frees, 1,064 bytes allocated
==31110==
==31110== 20 bytes in 1 blocks are definitely lost
```

```
==31110==    at malloc
==31110==    by allocate_array (memory_debug.c:8)
==31110==    by main (memory_debug.c:48)
==31110==
==31110== 20 bytes in 1 blocks are definitely lost
==31110==    at malloc
==31110==    by duplicate_array (memory_debug.c:34)
==31110==    by main (memory_debug.c:52)
==31110==
==31110== LEAK SUMMARY:
==31110==    definitely lost: 40 bytes in 2 blocks
==31110==
==31110== ERROR SUMMARY: 2 errors from 2 contexts
```

2.3 Key Findings

- **Total leaked:** 40 bytes (2 blocks of 20 bytes each)
- **Leak 1:** Original array allocated at line 48
- **Leak 2:** Duplicated array allocated at line 51
- **Root cause:** No `free()` calls for allocated memory

3 Solution Implementation

3.1 Fix 1: Implement free_memory()

Listing 1: Fixed free_memory() Function

```c
void free_memory(int *arr) {
    if (arr) {          // NULL check for safety
        free(arr);      // Release memory
    }
}
```

3.2 Fix 2: Free array_copy

Listing 2: Fixed main() Function

```c
int main() {
    int *array = allocate_array(SIZE);
    initialize_array(array, SIZE);
    print_array(array, SIZE);

    int *array_copy = duplicate_array(array, SIZE);
    print_array(array_copy, SIZE);

    free_memory(array);         // Free original array
    free_memory(array_copy);    // Free duplicated array - FIX

    return 0;  // No memory leaks!
}
```

4 Results - After Fix

4.1 Valgrind Output - After Fix

```
==32695== HEAP SUMMARY:
==32695==     in use at exit: 0 bytes in 0 blocks
==32695==   total heap usage: 3 allocs, 3 frees, 1,064 bytes allocated
==32695==
==32695== All heap blocks were freed -- no leaks are possible
==32695==
==32695== ERROR SUMMARY: 0 errors from 0 contexts
```

4.2 Comparison

Table 2: Before vs After Comparison

Metric	**Before Fix**	**After Fix**
Memory leaked (bytes)	40	0
Memory leaked (blocks)	2	0
Total allocations	3	3
Total frees	1	3
Valgrind errors	2	0
Code lines modified	0	2