**Exercise 1: Creating a Memory Leak**

Create a C program that intentionally causes a memory leak.

**Code:**

c

#include <stdio.h>

#include <stdlib.h>

void memory\_leak() {

int \*ptr = (int\*) malloc(sizeof(int) \* 10);

// Memory is allocated but never freed.

}

int main() {

memory\_leak();

printf("Memory leak example completed.\n");

return 0;

}

**Steps:**

1. Compile the program:

bash

gcc -o mem\_leak mem\_leak.c

**Exercise 2: Fixing the Memory Leak**

Modify the previous code to properly free the allocated memory.

**Code:**

c

#include <stdio.h>

#include <stdlib.h>

void memory\_leak\_fixed() {

int \*ptr = (int\*) malloc(sizeof(int) \* 10);

if (ptr == NULL) {

printf("Memory allocation failed.\n");

return;

}

// Free allocated memory

free(ptr);

}

int main() {

memory\_leak\_fixed();

printf("Fixed memory leak.\n");

return 0;

}

**Steps:**

1. Compile the program:

bash

gcc -o mem\_fixed mem\_fixed.c

**Exercise 3: Double Free Error**

Create a scenario where memory is freed twice.

**Code:**

c

#include <stdio.h>

#include <stdlib.h>

void double\_free() {

int \*ptr = (int\*) malloc(sizeof(int) \* 5);

if (ptr == NULL) {

printf("Memory allocation failed.\n");

return;

}

free(ptr);

free(ptr); // Double free error

}

int main() {

double\_free();

return 0;

}

**Run Valgrind and Observe Errors**

bash

gcc -o double\_free double\_free.c

**Exercise 4: Memory Leak in Loops**

A program that allocates memory inside a loop but forgets to free it.

**Code:**

c

#include <stdio.h>

#include <stdlib.h>

void memory\_leak\_loop() {

for (int i = 0; i < 5; i++) {

int \*ptr = (int\*) malloc(sizeof(int) \* 10);

// Memory is allocated in each iteration but never freed.

}

}

int main() {

memory\_leak\_loop();

return 0;

}

**Fix:**

Modify the loop to free memory properly:

c

void memory\_leak\_loop\_fixed() {

for (int i = 0; i < 5; i++) {

int \*ptr = (int\*) malloc(sizeof(int) \* 10);

if (ptr != NULL) {

free(ptr);

}

}

}

**Best Practices to Prevent Memory Leaks**

1. **Always free memory**: Every malloc should have a corresponding free.
2. **Use smart pointers** (if using C++ or managed environments).
3. **Set pointers to NULL** after freeing them:

c

free(ptr);

ptr = NULL;

1. **Use Valgrind regularly** for detecting memory leaks.
2. **Avoid memory allocation inside loops** unless necessary.
3. **Use static analysis tools** like clang-analyzer or cppcheck.