IC221 Lab: Memory Leaks Worksheet Name Umberto Fontana
Sping AY2022, 100 points total

Task 1 (50 points)

(5) Compile and execute `memleak.c`. Verify the output and review the program.

(10) Run valgrind on the memleak program. How many bytes does it say have been "definitely" lost?

60 bytes in 2 blocks.

(5) On what line(s) of code does valgrind indicate a memory leak has occurred? 34, 19, 50

(10) Identify and describe at least one memory leak in `memleak.c`.

Line 19: memory is allocated with calloc and assigned to the pointer new_a, but that memory never gets freed.
Line 34: memory is allocated with calloc and assigned to the pointer a. But that pointer gets overwritten in line 50, because it now points to the return value of the doubleup function. Thus, the memory that it was first referencing (with the first calloc call) is not freed and can never be freed: it's lost.

(10) Fix the memory leak you identified and verify your fix with `valgrind`.

(10) Describe how you fixed the memory leak:

I have initiated a new pointer to an int called b, and the return value of the doubleup() function, that is called in main, is now assigned to pointer b, so that pointer a is not overwritten. Then, at the end of the main function, I call free() on both a and b.

Task 2 (50 points)

(5) Compile and execute the `memviolation.c` program.

(10) Describe the output and execution of the program. Does it seem to be consistent?

In main, an int called i and a string (array of char) called hello are initialized. The string gets assigned the value “Hello World!”. Then, memory is allocated on the heap for a new string, which is the same size as hello and is called str. A for loop is then run, copying each character of hello in str, using indexes. Finally, the copied string is printed, and memory is freed. The output seems consistent.

(10) Run the program under valgrind. Identify the line of code that is causing the memory violation and its input:

Line 11 is causing the memory violation. We are making a call to malloc, assigning memory on the heap corresponding to strlen(hello).

(15) Describe the programming bug:

The issue here is the difference between the length of the string and the size of the string. The program is using strlen, which returns how many characters, not including the NULL character, are in the string. We should use the string size, which returns how many bytes are required to store the string. Then, when we copy the string, we should also copy the NULL character: the for loop should iterate until it is less-equal than strlen, not strictly less.

(10) Fix the memory violation and verify your fix with `valgrind`.

Submission

- Fixed `memleak.c`
- Fixed `memviolation.c`
- This completed worksheet