

# Exercises for Memory Tools

## Memory-Tools

### Uhoh

Continuing the example in the presentation.

1.

The memory allocation for names in `user.c` looks like this:

```
u->family_name = (char*) malloc(strlen(fname)*sizeof(char)+1);
u->given_name   = (char*) malloc(strlen(gname)*sizeof(char)+1);
```

If malloc fails, which you can fake by using a macro in `user.h`, we get NULL back which in turn is assigned to `u->family_name`. Trying to `strcpy` characters to that memory is, well... stupid!

So, fix the bug.

2.

Make sure there are no mem leaks.

*Hint: use valgrind*

3.

How should we signal to the calling function that we failed allocating memory?

4.

Assuming you've solved (1) and we now run our program. The `realloc` and the first `malloc` succeeds fails.

We now end up with memory for 1) one more user and 2) for one more name(string) but we're missing memory for the last name(string).

What do we do?

5.

Add code to do this.

## bad memories

10.

Create a C file with a main function. Call the file bad-memories.c

11.

The file should allocate (dynamically) memory for 100 ints.

*Note: do not deallocate (free) the memory*

12.

Compile the program and make sure it is called bad-memories.

13.

Make sure to use the following flags/options to gcc `-pedantic -Wall -Werror` when compiling.

You will get a warning about p is unused. Cool isn't it.

14.

Add a printf call to printout p. The code should compile after this since p is used (still a really useless program).

Run the program in Valgrind, like this:

`valgrind --leak-check=full bad-memories` What does Valgrind have to say about your program?

15.

Add `-g` to the list of gcc options. Recompile and execute via valgrind again (as in (4)).

What does Valgrind have to say about your program?

16.

Fix the memory problem.

*Note: we still want to use the memory so the allocation shall remain*

17.

Set the first int value to 7 and the 101:th to 12

Compile and run.

*Note: the code is faulty but may still work.*

18.

Run the code in Valgrind, like in (4).

What does Valgrind have to say about your program?

19.

Fix the problem and make sure Valgrind does not complain anymore.

*Note: remove the writing to the 101:th int*