Exercise Class 12 – Theory

1 Pointers and Memory Management

Correctly allocating and deallocating memory is a non-trivial task. This is especially a problem in large code bases where objects are often deallocated far away from the location in which they were created. Therefore, when dealing with dynamic memory it is useful to think in terms of ownership. The core idea of ownership is that a memory location can have only one owner at a time and that owner is responsible for deallocating it. (You can also have more, but then you need to figure out which is the last one.)

For example, the function `fib` presented in the slides of the exercise session allocates an array and transfers the ownership of that array to the caller. When `test2` calls `print`, it transfers the ownership of the array to it. When `print` finishes its work, we have a choice: either `print` deallocates the array or returns its ownership to the caller. In the latter case, `test2` is responsible for deallocating the array.

Memory can be owned not only by functions, but also by classes. For example, the stack shown in the lecture owns the dynamically allocated memory. Therefore, we should never deallocate that memory from outside of the stack.