What did we learn in Lab 02?

- What is the difference between deep and shallow copying?
- The compiler, by default, makes a copy constructor for every class. Is this constructor a deep or shallow copy constructor?
- What is the disadvantage of shallow copying?
- How does **ddd** work, and why is it a useful tool for debugging?
- Why is it important to have a clear mental picture of the problem before writing any code?
- Explain this situation. Where does 'b' point to?:

```cpp
int *a = new int;
int *b;
b = a;
delete a;
```

- What is statically allocated memory? What is dynamically allocated memory?
- Consider this code from the quiz:

```cpp
class Test{
public:
Test(int a) {
    p = new int;
    *p = a;
}

~Test() {
    cout << *p << "\n";
    delete p;
    p = NULL;
}
private:
    int *p;
};

int main() {
    Test t1(10);
    Test *t2 = new Test(20);
    return 0;
}
```

→ What is the output of this program?
→ Who is responsible for calling the destructor of t1 and t2?

- You are pair programming with a friend! Your friend has the driver role. Your friend is tired of coding for after an hour! What would you do?
What we are going to learn in Lab 03?

- Memory is leaking somewhere! Sooner or later we will run out of memory! Don't worry! Detective valgrind is here!
- What is the difference between allocating memory on the heap and allocating memory on the stack?
- What is the purpose of nullifying a pointer?
- What is the difference between delete[] and delete?
- Is there any problem (memory wise) with the following code?:

```cpp
int *a = new int;
int *b = a;
delete b;
```

- What is the significance of releasing the memory properly?