How's your weekend?

Mentimeter

birthday

warm and sunny

nice

2

Today's plan

- Review Ex 10-3
- Recap questions
- In-class Ex 11-1

Mentimeter

Ex 10-3: overload **operator<<**

- Can use friend keyword to allow access private members
- friend std::ostream& operator<<(std::ostream& os, const Complex& rhs);
- os << rhs.rel << " + " << rhs.img << "i";
- Otherwise, you can use the getter (you don't need friend access if using getter):
- os << rhs.getRel() << " + " << rhs.getImg() << "i";

Ex 10-3: overload **operator[=+- *](...)**, the copy constructor and the assignment operator

- Straightforward: implement the formula
- Copy constructor and assignment operator: copy it field by field.
- In fact, because it only has two primitive type fields, the default one will work as well
- We added them for practicing

Ex 10-3: multiply **float** on the left side?

- overload float times Complex operator
- use friend to allow the function to access Complex's private members
- friend Complex operator* (const float& lhs, const Complex& rhs);

Mentimeter

Summary of Ex 10-3

- Practiced how to overload class's operators
- Learned how to use `friend` keyword to allow other functions to access class's private members
- Notice that: `friend` functions declared is not a member function of the class (even though we put it in the class definition).
- If you don't want to use `friend`, you can always use `getter` and `setter` to achieve the same goal.

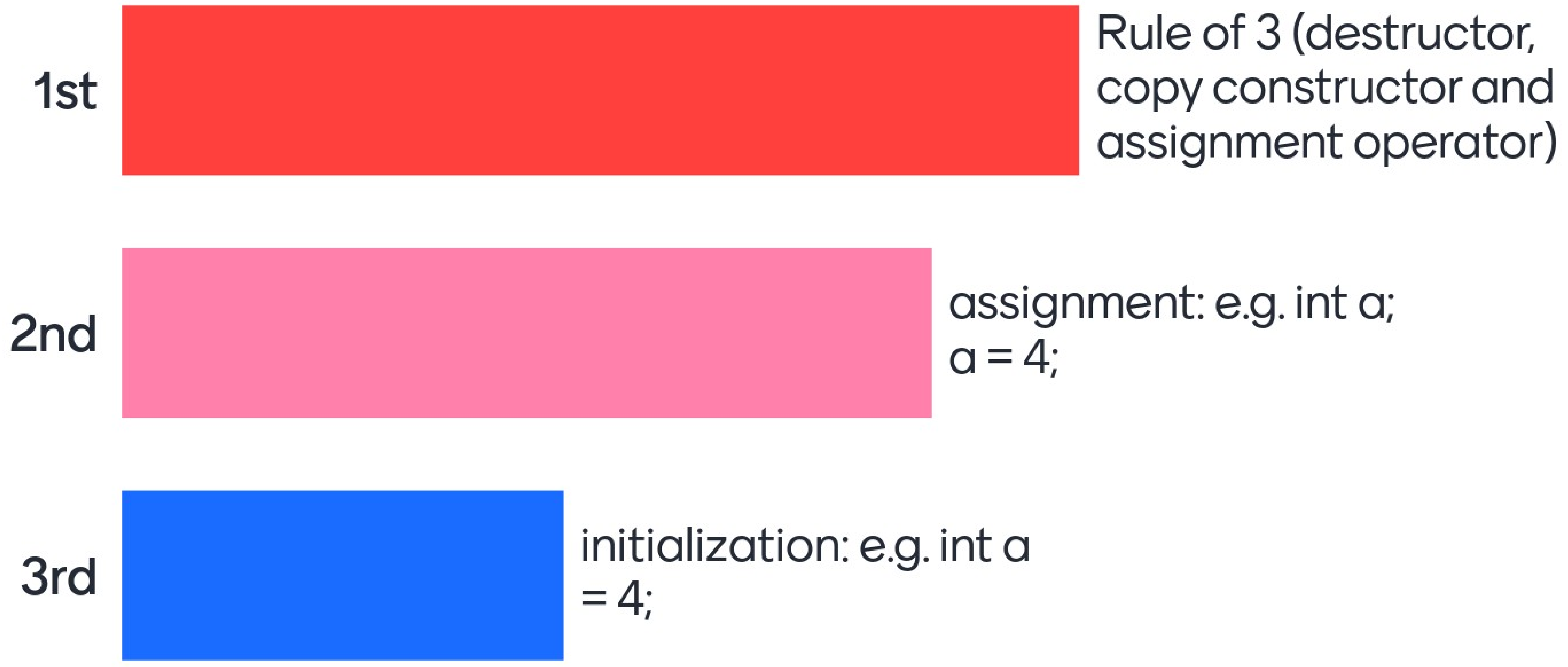
Mentimeter

More about **friend**, getter/setter, and public fields

- friend is always the best option because it has the strictest access
- if you start to have lots of getter/setter, think about to redesign your model.
- you probably want to friend the class that modifies the fields instead of having getter/setter
- OO is also about the data access. Try not to make your fields public
- If you have a getter/setter, no one can set a ref/pointer to your data
- But if you make it public, others can set a ref/pointer to the data and change it whenever they want instead calling your method

Rank by the unfamiliarity

Mentimeter



5

Mentimeter

Rule of 5 (C++11) [FYI only, not in exam]

- destructor: ~Class();
- copy constructor: Class(const Class&);
- (copy) assignment operator: Class& operator=(const Class& other);
- move constructor: Class(Class&&);
- move assignment operator: Class& operator=(Class&& other);

Mentimeter

move semantics in C++11 [FYI only, not in exam]

- It allows to transfer/take ownership of memory
- Class&& is called rvalue reference
- rvalue: value that is extracted from memory or a constant
- Move constructor: swap the content, then destroy the old object
- e.g. class A { int len; int* data; };
- A(const A& other) : len(other.len), data(new int[other.len]) {// copy other.data to data}
- A(A&& other) : int(other.len), data(other.data) {other.len = 0; other.data = nullptr;} }

Mentimeter

move semantics in C++11 [FYI only, not in exam]

- e.g. class A { int len; int* data; };
- A& operator=(const A& other) { if (this != other) { len = other.len; data = new int[len]; // copy data logic;} return *this; }
- A& operator=(A&& other) { if (this != other) { if (data) delete[] data; len = other.len; data = other.data; other.data = nullptr;} return *this; }
- In sum, in C++11, if you have non-default destructor, you should define copy, move constructor and assignment operator.

What is the difference between initialization and assignment?

Initialization calls the copy constructor while assignment calls the assignment operator

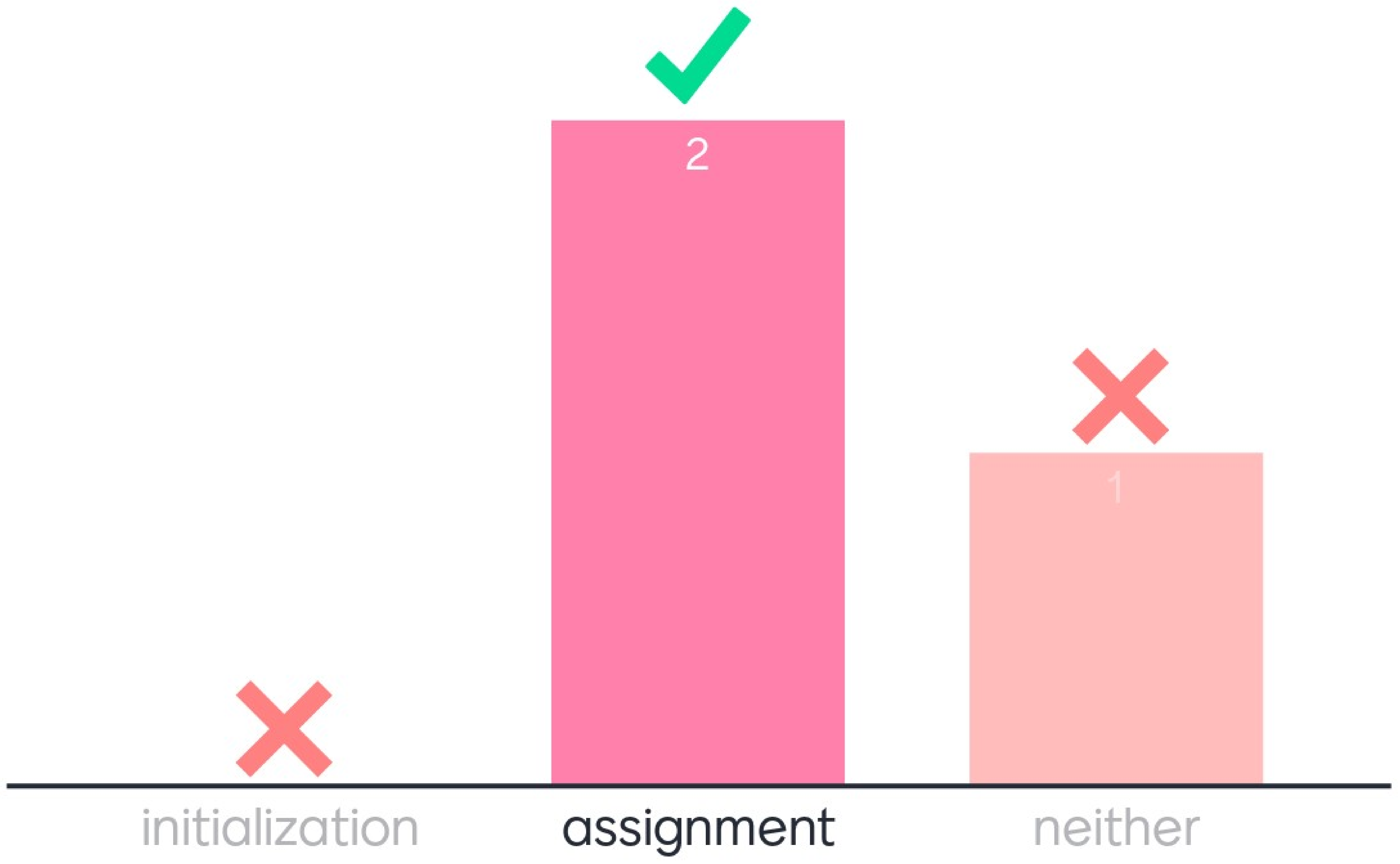
initialization calls the constructor, assignment calls the copy funciton

The correct answer is: initialization happens during declaration and it calls the appropriate constructor. assignment calls the assignment operator.

2

Does the line **f2 = f1;** use initialization or assignment (assume **Foo** is a class and **f1** and **f2** are both of type **Foo**)?

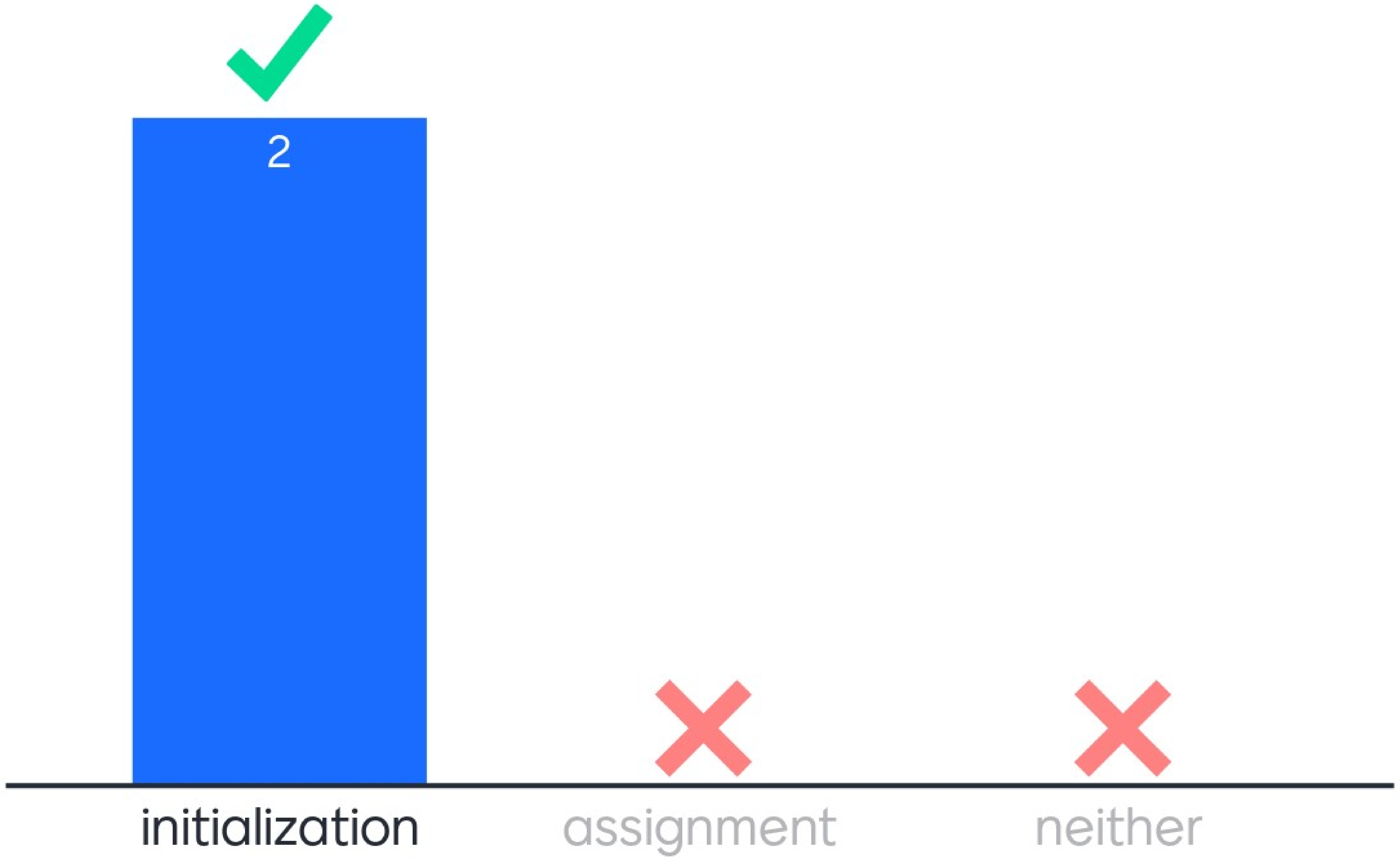
Mentimeter



3

Does the line **Foo f2 = f1;** use initialization or assignment (assume **Foo** is a class and **f1** is of type **Foo**)?

Mentimeter



2

Mentimeter

What is the difference between shallow and deep copy?

shallow copy will copy references to created memory, but not create new memory with the same values. ✗

shallow copy have the same data and points to the same memory location ✗

The correct answer is: Shallow only copies pointers while deep allocates memory and copies over values.

2

What is the rule of 3 (or 5 in c++11)?

When you have a destructor to implement, you have to implement your own copy constructor and overload the assignment operator. ✕

if you a deleting memory, define assignment, copy, move ✕

we should modify destructor, assignment operator, and copy constructor at the same time ✕

The correct answer is: If you have defined/deleted your own destructor, copy/move constructor, or copy/move assignment, you should define them all!

4

Ask me anything

Mentimeter

0 questions
0 upvotes