# Today's plan

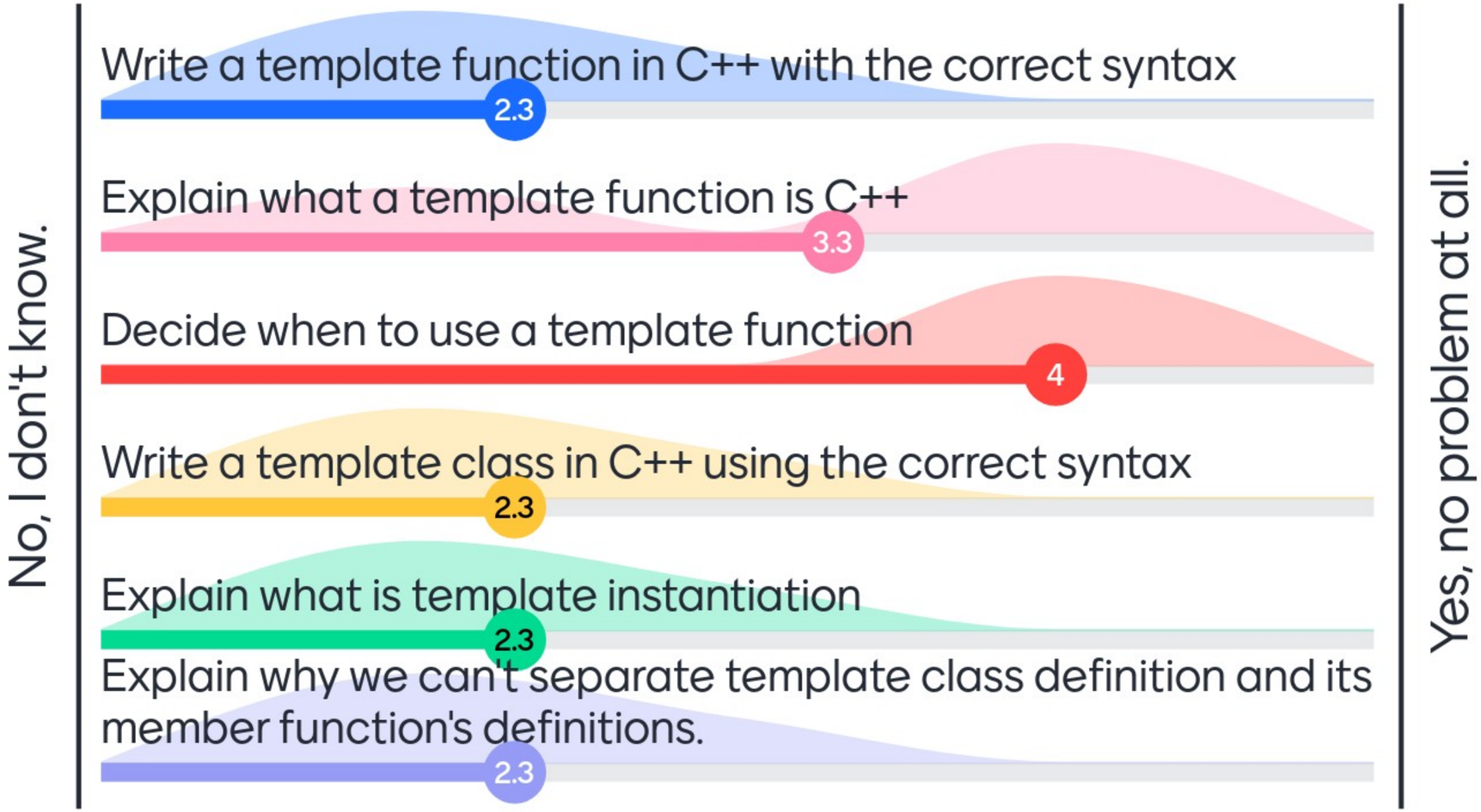
- → Review Ex 11-1
- → Recap questions
- → In-class Ex 11-2

# Ex 11-1: solutions is available for you to extend it to templated class (Ex 11-2)

- If you do it right, the only logic you need to implement is to traverse the set using the get_next() function..
- Copy constructor: call the default constructor in the initializer list, in the body, traverse the set and use add().
- Destructor: call clear()
- operator+=(): call add()
- operator=(): check if it is a self-assignment, if not, clear, then traverse the set and add
- operator<<(): traverse the set and print

# Do you know how to do this in C++?

Mentimeter



# Write a simple template function declaration in proper C++ syntax.

Mentimeter

void print(T string){}

The correct answer is: template< typename T > func() { // use T in the body or a the parameter type};

1

# Under what conditions would you consider making a function templated?

Mentimeter

- If the body of a function is identical except for the types of data you are using, you should make the function templated. ✕
- when we would overload a function many times for different types. ✕
- when the same body can work for multiple variable types. ✕
- when the body of the function are essentially the same, only the datatype of parameter varies ✕

The correct answer is: Same function body (logic), but with different types.

4

# What is template instantiation?

Mentimeter

- The compiler doesn't instantiate templates until first use and it needs template classes and functions to be defined already. ✗
- since the compiler is lazy, it will instantiate templates when it is called for the first time ✗
- the compiler will compile when it firstly meets a template function ✗

The correct answer is: When the template function/class first uses with a specify type, the compiler will instantiate a concrete one from the template for that type.

3

# Can we separate declaration and definition when using templates? If so, how?

Mentimeter

No we need to define the template class in header files (can't separate declaration and definition into .h and .cpp files)

write definitions in some other file like, .inc

include .inc in the header file

There's no correct answer!

3

# Why shouldn't template definitions be in .cpp files?

Mentimeter

- By convention, we don't #include .cpp files so can't have template definitions in .cpp ✗
- since the compiler will not see the definitions when instantiating for the first time ✗
- because the instantiation need to see the full definition ✗

The correct answer is: Because when the compiler instantiate the template, it needs to see the full definition.

3

# Ask me anything

Mentimeter

0 questions

0 upvotes