Exercise 1: Template with same type parameters

Create a template function that takes in two parameters and returns the maximum of both.

In main, test your template function with different types of arguments. Use both implicit and explicit template instantiation.

Exercise 2: Template with different type parameters

Create a template function that takes in two parameters (of different types) by reference and swaps their values.

(Hint: template <typename T, typename U>)

In main, test your template function. Use both implicit and explicit template instantiation.

Exercise 3: Class with Public Access

Define a **Product** class with the following spec:

✓ **Public** Data Members: **productID** (string), **name** (string),
price (double)

Test your class in main with two objects and use the dot operator to initialize with following *(For example, p1.name = "Notepad")*

Object 1: (productID: **P001**, name: **iPad**, price: **849.99**)

Object 2: (productID: **P003**, name: **Electric Kettle**, price: **24.99**)

Finally, print each object's data members using the dot operator.

Exercise 4: Class with Private Access (Setters & Getters)

Define an **Animal** class with the following spec:

✓ **Private** Data Members: **name** (string), **age** (int), **type** (string), **isPet** (bool)

✓**Public** Methods:

1. Setter methods for all four data members
2. Getter methods for all four data members

Test your code in main with two objects and use the setter methods to initialize with following:

Object 1: (name: **Whiskers**, age: **3**, type: **Cat**, isPet: **true**)

Object 2: (name: **Buddy**, age: **5**, type: **Dog**, isPet: **true**)

Print each object's four data members using the getter methods.

Exercise 5: Class with Default Constructor

Define a **Movie** class with the following spec:

✓ **Private** Data Members: **title** (string), **director** (string),
rating (double), **duration** (int)

✓**Public** Methods:

1. Default Constructor
   * initializes the **title** to "Inception"
   * initializes the **director** to "Christopher Nolan"
   * initializes the **rating** to 8.8
   * initializes the **duration** to 148
2. Two Setter methods (one for **title**, one for **director**)

Continue on Next Slide…

Exercise 5: Class with Default Constructor

✓**Public** Methods:

3. A Display method that prints a movie's attributes

In main:

* Create two objects (*m1* and *m2*)
* Use the display method to print out their attributes
* Using the setter methods, modify the **title** and **director** of *m2* to "Buddy Buddy" and "Billy Wilder" respectively
* Confirm this modification by displaying *m2's* attributes

Exercise 6: Class with Default Constructor (Initializer Lists)

Modify the default constructor in Exercise 5 to use an initializer list.

Exercise 7: Class with Parameterized Constructor

Define a **VectorFiller** class with the following spec:

✓ **Private** Data Members: **values** (a vector of integers)

✓**Public** Methods:

1. Default Constructor
    * initializes **values** with 10 elements, each set to 0.
2. Parameterized Constructor
    * takes an integer *n* as a parameter
    * initializes **values** with 10 elements, each set to *n*
3. A Print method that displays the contents of **values**

Continue on Next Slide…

Exercise 7: Class with Parameterized Constructor

In main:

* Create object *vf1* that calls default constructor
* Create object *vf2* that calls parameterized constructor
  - ✓Pass *-1* as the argument to the parameterized constructor
* Using the Print method, display the contents of both vectors *(If you did it right, you'll have 10* ***0's*** *and 10* ***-1's*** *in your output!)*

Exercise 8: Class using Multiple Files

Break your code for Exercise 7 into multiple files:

* **VectorFiller.h**: This file contains the class declaration and method prototypes
* **VectorFiller.cpp**: This file contains the implementation of all the methods of your class (including constructors)
* **main.cpp**: This is your driver file and should contain the main function

1. You must include the .h file in all .cpp files as follows:

```
#include "VectorFiller.h"
```

2. Compile **only** .cpp files: **g++ VectorFiller.cpp main.cpp -o ex8**

True or False?

1. Default constructors are automatically provided by the compiler when one or more constructors are defined explicitly in a class

2. Dot notation is used to access member variables and methods of an object in C++

3. A class member declared as *protected* in C++ is accessible from outside the class through dot notation

4. Default constructors take at least one argument

5. Setters and getters are not necessary if all class members are declared as *public*

True or False?

6. In certain cases, constructors in C++ may have a return value

7. A data member declared as *private* can be accessed by a member function of the same class

8. A member function declared as *protected* is inaccessible using dot notation in the main function

9. Every instance of a class is an object

10. A constructor can be invoked/called explicitly as long as it is declared as *public*

11. It is possible to have all three access specifiers in one class