Structured Bindings and Custom Types Solutions

Function which Returns Different Types

- Is it possible to have a C++ function which returns different types in different branches?
    - Not possible up to C++11 (except for polymorphic types)
    - This became possible in C++14, where the function return type can be "auto"
    - However, the function must return the same type in each branch
    - Template metaprogramming can be used to exclude the unused branches from the instantiation, leaving only the branch which is taken
    - This can be done much more easily in C++17 with constexpr if
    - In C++17, the function could also return std::variant instead (although this makes changing the returned types harder)

Structured Bindings and Custom Types

- Write a class
- Use constexpr if to implement a get<> member function for your class
- Write a function which returns an object of this class
- Write a program which uses a structured binding to unpack the elements of the returned class

Implementing tuple_size and tuple_element

- (Optional)
- Modify your solution so that it displays the last member (in declaration order) of the class
- Make sure that your solution safely handles the case of a class with no members