Advanced Programming in C++
Aggregates and list initialization

An *aggregate* is an *array* or a *class* with no user-provided constructors, no brace-or-equal-initializers for non-static data members (this may be allowed in C++14), no private or protected non-static data members, no base classes, and no virtual functions. Examples of brace-or-equal- initializer are:

```
int i{1};      // brace initializer
int i = 1;     // equal initializer
```

An aggregate can be initialized by an *initializer list*, so called *list initialization*, that is, can be initialized from a *braced-init-list*. Below are two examples of braced-init-lists:

```
{ 1, 2, 3 }
{ "Foo", 17, 3,14, true }
```

The comma-separated *initializer-clauses* of an initializer list are called the *elements* of the list. An initializer list may be empty, `{}`. List-initialization can occur in both direct-initialization or copy-initialization contexts:

```
T x{a};     // direct-initialization syntax
T x(a);     // direct-initialization syntax
T x = a;    // copy-initialization syntax
```

List-initialization is important to know about and can be used in many contexts:

1) as the initializer in a variable definition

2) as the initializer in a new expression

3) as a function argument

4) in a return statement

5) as an initializer for a non-static data member

6) in a member initializer

7) on the right-hand side of an assignment

8) as an argument to a constructor invocation

9) as a subscript

Do the following for 1–8 above:

a) write test cases when relevant for array

b) write test cases when relevant for **struct** aggregate

c) write test cases for context where array or **struct** aggregate gave problems

Hint: std::initializer_list may be of interest in some cases where array is not possible.