Swinburne University of Technology

School of Science, Computing and Engineering Technologies

LABORATORY COVER SHEET

Subject Code:	COS30008
Subject Title:	Data Structures and Patterns
Lab number and title:	10, Copy and Move Semantics & Emplace
Lecturer:	Dr. Markus Lumpe

Any sufficiently advanced technology is indistinguishable from magic.
Arthur C. Clarke

Lab 10: Copy and Move Semantics & Emplace

Consider the following template class `Array`:

```
#pragma once

#include <cstdint>
#include <algorithm>
#include <cassert>

template<typename T>
class Array
{
public:

  Array() noexcept;
  ~Array();

  // copy semantics
  Array( const Array& aOther );
  Array& operator=( const Array& aOther );

  // move semantics
  Array( Array&& aOther ) noexcept;
  Array& operator=( Array&& aOther ) noexcept;

  void swap( Array& aOther ) noexcept;

  // array operations
  size_t size() const noexcept;

  const T& operator[]( size_t aIndex ) const noexcept;

  void reserve( size_t aNewSize );

  void fill( const T& aValue = T{} ) noexcept;

  template<typename... Args>
  void emplace_at( size_t aIndex, Args&&... args ) noexcept;

private:

  T* fElements;
  size_t fSize;
};
```

Define the `Array` template class using the techniques we have studied in the unit.

The methods `reserve()` and `fill()` have the expected semantics. The `reserve()` method allocates `aNewSize` space for the array. The content of the old array must be moved into the new space. The old space must be freed at the end. The `fill()` method copies the parameter value into every array element.

The `emplace_at()` method creates an object of type `T` at `aIndex`. The insertion point must be within the array. Use perfect forwarding to allow the compiler to choose the best-matching overloaded constructor for type `T`.

Use `Main.cpp` to test your implementation. The main program must not throw a runtime exception. It should produce the following output:

```
Start Array test...
To , length = 3
be, , length = 4
or , length = 3
```

```
not , length = 4
to , length = 3
be: , length = 4
that , length = 5
is , length = 3
the , length = 4
question:, length = 9
lArray1 size: 10
To be, or not to be: that is the question:
To be, or not to be: that is the question:
lArray1 size: 10
lArray2 size: 0
lArray3 size: 0
To be, or not to be: that is the question:
To be, or not to be: that is the question:
All arrays go out of scope.
Array test complete.
```