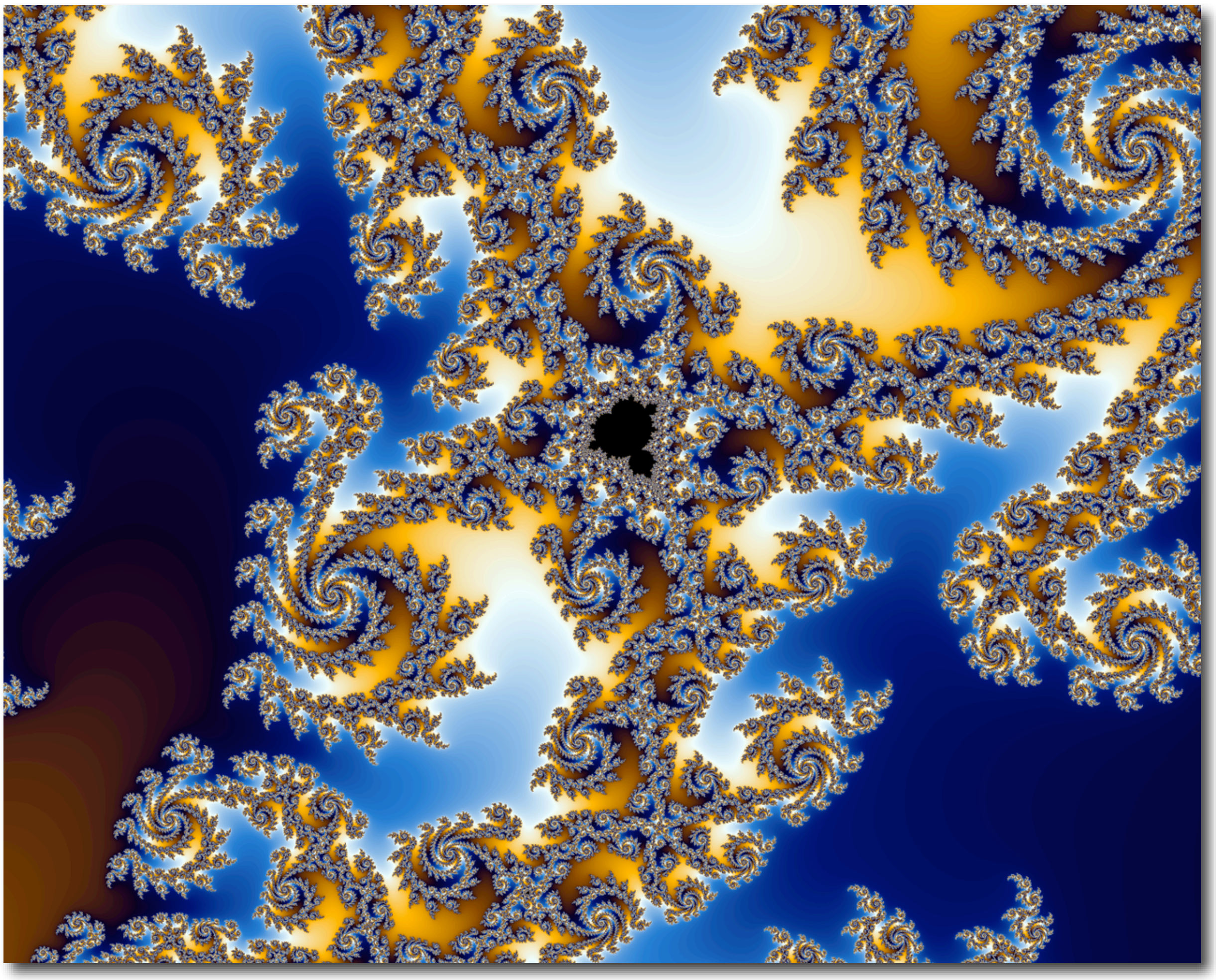
Swinburne University Of Technology

Faculty of Information and Communication Technologies

LABORATORY COVER SHEET

Subject Code:	COS30008
Subject Title:	Data Structures and Patterns
Lab number and title:	4, Iterators – Problem 1 Solution Steps
Lecturer:	Dr. Markus Lumpe



Problem 1 – Solution Approach

1. Create a new project and add class `CharacterCounter` and `main` function.

   Refactor `CharacterCounter` as a new class. This should improve code organization. Use the solution from tutorial 3 as input. We need to add a new function to allow access to a given character frequency:

   ```
   int operator[]( unsigned char aCharacter ) const;
   ```

   The reason: all member variables are private and we need access. However, as part of good OO practice, we want to preserve object-oriented encapsulation.

   Build project. There should be no errors.

2. We need a frequency map to associate a character with its number of occurrences in the iterator. We use the class `FrequencyMap` for this purpose. Add a new class `FrequencyMap` to the project, implement its behavior, and rebuild the project. There should be no errors.

3. Add new class `CharacterCounterIterator` to the project. Use the lecture example IntArrayIterator as a prototype for the class implementation.

   Key points:

   - Constructor: we need to initialize the `fMap` array using the `CharacterCount` object `aCounter`. The interfaces are quit different. To create an object in C++, we can call a suitable constructor:

     ```
     FrequencyMap( i, aCounter[ i ] ).
     ```

     Position member variable `fIndex` on the first non-zero frequency map.

   - operator++: Position member variable `fIndex` on the first non-zero frequency map.

   - operator++(int): Do not reinvent the wheel. Recycle existing operators, that is, implements the postfix operator in terms of the prefix operator: `++(*this)`.

   - operator==: How do we determine equivalence? Answer, same index and same underlying collection. But there is a problem; it is not feasible to test the underlying collection. Solution, observe iterator context, that is, we only compare iterators originating (statically) from the same character counter. Hence, we only need to check the position.

   - begin(): Problem, we do not have access to the original character counter. Solution, create a copy and set the copy's `fIndex` to first non-zero frequency map;

   - end(): As begin(), but use 256.

   Build project. There should be no errors.

4. Add iterator to main function and rebuilt.

5. Test program with Main.cpp. Console output should be similar to the one shown in the tutorial notes.