Swinburne University of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code:	COS30008
Subject Title:	Data Structures & Patterns
Assignment number and title:	2 - Iterators
Due date:	Monday, 22 April, 2024, 10:30
Lecturer:	Dr. Markus Lumpe

Your name: Nur E Siam **Your student id:** 103842784

Marker's comments:

Problem	Marks	Obtained
1	40	
2	70	
Total	110	

Extension certification:

This assignment has been given an extension and is now due on ____________

Signature of Convener:__________________________

```cpp
// COS30008
// Created by Nur E Siam

#include "FibonacciSequenceGenerator.h"
#include <cassert>

// Constructor initializes the Fibonacci sequence generator
FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string& 
  aID) noexcept
    : fID(aID), fPrevious(0), fCurrent(1) {}

// Getter for the generator ID
const std::string& FibonacciSequenceGenerator::id() const noexcept {
    return fID;
}

// Dereference operator overload to retrieve the current Fibonacci number
const long long& FibonacciSequenceGenerator::operator*() const noexcept {
    return fCurrent;
}

// Conversion operator to bool to check if there are more Fibonacci numbers
FibonacciSequenceGenerator::operator bool() const noexcept {
    return hasNext();
}

// Reset the generator to the initial state
void FibonacciSequenceGenerator::reset() noexcept {
    fPrevious = 0;
    fCurrent = 1;
}

// Check if there are more Fibonacci numbers in the sequence
bool FibonacciSequenceGenerator::hasNext() const noexcept {
    return fCurrent <= LLONG_MAX - fPrevious;
}

// Generate the next Fibonacci number in the sequence
void FibonacciSequenceGenerator::next() noexcept {
    long long temp = fCurrent;
    fCurrent += fPrevious;
    fPrevious = temp;
}
```

```cpp
// COS30008
// Created by Nur E Siam

#include "FibonacciSequenceIterator.h"

// Constructor for Fibonacci sequence iterator
FibonacciSequenceIterator::FibonacciSequenceIterator(const
  FibonacciSequenceGenerator& aSequenceObject,
    long long aStart) noexcept
    : fSequenceObject(aSequenceObject), fIndex(aStart) {}

// Dereference operator to retrieve the current Fibonacci number
const long long& FibonacciSequenceIterator::operator*() const noexcept {
    return *fSequenceObject;
}

// Pre-increment operator to move to the next Fibonacci number
FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept
   {
    fSequenceObject.next();
    ++fIndex;
    return *this;
}

// Post-increment operator to move to the next Fibonacci number
FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int)
  noexcept {
    FibonacciSequenceIterator temp = *this;
    ++(*this);
    return temp;
}

// Equality operator to check if two iterators point to the same index
bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator&
   aOther) const noexcept {
    return fIndex == aOther.fIndex;
}

// Inequality operator to check if two iterators point to different indices
bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator&
   aOther) const noexcept {
    return fIndex != aOther.fIndex;
}

// Get the iterator pointing to the beginning of the sequence
FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept
   {
    return FibonacciSequenceIterator(fSequenceObject, 1);
}
```

```cpp

// Get the iterator pointing to the end of the sequence
FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
    return FibonacciSequenceIterator(fSequenceObject, 93);
}

```