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: H M Asfaq Ahmed Shihab **Your student id:** 103849248

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
//
//  FibonacciSequenceGenerator.cpp
//  problem_Set2
//
//  Created by H M Asfaq Ahmed Shihab on 18/4/2024.

#include "FibonacciSequenceGenerator.hpp"
#include <cassert> // For assertion

FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string&
 aID) noexcept
    : fID(aID), fPrevious(0), fCurrent(1) {}

const std::string& FibonacciSequenceGenerator::id() const noexcept {
    return fID;
}

const long long& FibonacciSequenceGenerator::operator*() const noexcept {
    return fCurrent;
}

FibonacciSequenceGenerator::operator bool() const noexcept {
    return hasNext();
}

void FibonacciSequenceGenerator::reset() noexcept {
    fPrevious = 0;
    fCurrent = 1;
}

bool FibonacciSequenceGenerator::hasNext() const noexcept {

    return fCurrent <= LLONG_MAX - fPrevious;
}

void FibonacciSequenceGenerator::next() noexcept {

    if (!hasNext()) {

        return;
    }

    long long next = fPrevious + fCurrent;
    fPrevious = fCurrent;
    fCurrent = next;
}
```

```cpp
//
//  FibonacciSequenceIterator.cpp
//  problem_Set2
//
//  Created by H M Asfaq Ahmed Shihab on 18/4/2024.
#include "FibonacciSequenceIterator.hpp"

FibonacciSequenceIterator::FibonacciSequenceIterator(const
 FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept
    : fSequenceObject(aSequenceObject), fIndex(aStart - 1) {}

const long long& FibonacciSequenceIterator::operator*() const noexcept {
    return fSequenceObject.operator*();
}

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

FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int)
 noexcept {
    FibonacciSequenceIterator temp(*this);
    ++(*this);
    return temp;
}

bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator&
 aOther) const noexcept {
    return (fSequenceObject.id() == aOther.fSequenceObject.id()) && (fIndex
     == aOther.fIndex);
}

bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator&
 aOther) const noexcept {
    return !(*this == aOther);
}

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept
 {
    return FibonacciSequenceIterator(fSequenceObject, 1);
}

FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
    return FibonacciSequenceIterator(fSequenceObject, 93);
}
```