Swinburne University of Technology

School of Science, Computing and Emerging Technologies

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures & Patterns
Assignment number and title: 2 - Iterators
Due date: Sunday, 13 April, 2025, 23:59
Lecturer: Dr. Markus Lumpe

Your name: Xinzhe Yu **Your student ID:** 105385294

Marker's comments:

Problem	Marks	Obtained
1	44	
2	64	
Total	108	

Extension certification:

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

Signature of Convener:_____

```cpp
//
//  FibonacciSequence.cpp
//  problemset2
//
//  Created by Xinzhe Yu on 6/4/2025.
//

#include "FibonacciSequence.h"

FibonacciSequence::FibonacciSequence() noexcept:
    fPrevious(0),
    fCurrent(1)
{}

const uint64_t& FibonacciSequence::operator*() const noexcept
{
    return fCurrent;
}

FibonacciSequence& FibonacciSequence::operator++() noexcept
{
    fCurrent = fCurrent + fPrevious;
    fPrevious = fCurrent - fPrevious;
    return *this;
}

FibonacciSequence FibonacciSequence::operator++(int) noexcept
{
    FibonacciSequence temp = *this;
    // go to the next
    ++(*this);
    // but return the previous one
    return temp;
}

bool FibonacciSequence::operator==(const FibonacciSequence& aOther) const
 noexcept
{
    return fCurrent == aOther.fCurrent && fPrevious == aOther.fPrevious;
}

void FibonacciSequence::begin() noexcept
{
    fPrevious = 0;
    fCurrent = 1;
}

void FibonacciSequence::end() noexcept
{
    fPrevious = 0;
    fCurrent = 0;
}
```

```cpp
//
//  FibonacciSequenceIterator.cpp
//  problemset2
//
//  Created by Xinzhe Yu on 7/4/2025.
//
#include <iostream>
#include "FibonacciSequenceIterator.h"

FibonacciSequenceIterator::FibonacciSequenceIterator(FibonacciSequence*
 aSequence, uint64_t aStart) noexcept:
    fSequence(aSequence),
    fIndex(aStart)
{
//    if the aSequence is existing, let the aSquence iterate to the
 position of fIndex, if no fIndex, it will start from 0
    if (fSequence != nullptr)
    {
        fSequence->begin();
        for (uint64_t i = 1; i < fIndex; ++i)
        {
            ++(*fSequence);
        }
    }
}

const uint64_t& FibonacciSequenceIterator::operator*() const noexcept{
    return **fSequence;
}

FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept{
        ++(*fSequence);
        ++fIndex;
    return *this;
}

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

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

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept
{
    return FibonacciSequenceIterator(fSequence, 0);
}
```

```
FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept
{
    return FibonacciSequenceIterator(fSequence, MAX_FIBONACCI);
}
```