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: Dhanveer Ramnauth **Your student ID:** 103866373

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:_____

Figure 1: FibonacciSequence.cpp

```cpp
#include "FibonacciSequence.h"
#include <cstdint>

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

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

FibonacciSequence& FibonacciSequence::operator++() noexcept
{
    uint64_t next = fPrevious + fCurrent;
    fPrevious = fCurrent;
    fCurrent = next;
    return *this;
}

FibonacciSequence FibonacciSequence::operator++(int) noexcept
{
    FibonacciSequence old = *this;

    ++(*this);

    return old;
}

// Iterator must implement operator==. The operation != is synthesized.
bool FibonacciSequence::operator==( const FibonacciSequence& aOther ) const noexcept
{
    return fCurrent == aOther.fCurrent &&
           fPrevious == aOther.fPrevious;
}


```

```
39 void FibonacciSequence::begin() noexcept
40 {
41     fPrevious = 0;
42     fCurrent  = 1;
43 }
44
45 void FibonacciSequence::end() noexcept
46 {
47     fPrevious = 0;
48     fCurrent = 0;
49 }
```

Figure 2: FibonacciSequenceIterator.cpp

```cpp
#include "FibonacciSequenceIterator.h"

FibonacciSequenceIterator::FibonacciSequenceIterator(
    FibonacciSequence* aSequence, uint64_t aStart) noexcept
    : fSequence(aSequence), fIndex(aStart)
{
    if (fSequence && aStart == 0)
    {
        fSequence->begin();
    }
    else if (fSequence)
    {
        fSequence->begin();
        for (uint64_t i = 0; i < aStart; ++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 old = *this;

    ++(*this);

    return old;
```

```
37 }
38
39 bool FibonacciSequenceIterator::operator==( const
   ↪ FibonacciSequenceIterator& aOther ) const noexcept
40 {
41     return fIndex == aOther.fIndex;
42 }
43
44 FibonacciSequenceIterator FibonacciSequenceIterator::begin()
   ↪ const noexcept
45 {
46     return FibonacciSequenceIterator(fSequence, 0);
47 }
48
49 FibonacciSequenceIterator FibonacciSequenceIterator::end() const
   ↪ noexcept
50 {
51     return FibonacciSequenceIterator(fSequence, MAX_FIBONACCI);
52 }
```