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: Rox Nguyen **Your student id:** 103500095

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
#include "FibonacciSequenceGenerator.h"
#include <limits>
#include <stdexcept>

// Constructor to set up a Fibonacci sequence
//sequence ID + prev 0 + cur 1 as required
FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string& aID) noexcept
    : fID(aID), fPrevious(0), fCurrent(1) {}

//Get sequence ID
const std::string& FibonacciSequenceGenerator::id() const noexcept
{
    return fID;
}

//Get current Fibonacci number
const long long& FibonacciSequenceGenerator::operator*() const noexcept
{
    return fCurrent;
}

//Type conversion to bool
FibonacciSequenceGenerator::operator bool() const noexcept
{
    //if current positive + next number can be calculated (same as hasNext())
    return fCurrent > 0 && std::numeric_limits<long long>::max() - fPrevious >= fCurrent;
}

//Reset sequence generator to first Fibonacci number
void FibonacciSequenceGenerator::reset() noexcept
{
    fPrevious = 0;
    fCurrent = 1;
}

//Tests if there is a next Fibonacci number
bool FibonacciSequenceGenerator::hasNext() const noexcept
{
    return std::numeric_limits<long long>::max() - fPrevious >= fCurrent;
}

//next Fibonacci number
void FibonacciSequenceGenerator::next() noexcept
{
    long long next = fCurrent + fPrevious;
    if (next < fCurrent) {
        //overflow assertion check
        throw std::overflow_error("Overflow in Fibonacci number calculation");
    }
    fPrevious = fCurrent;
    fCurrent = next;
}

```

```cpp
constexpr long long MAX_FIBONACCI = 92; //to have a fixed end limit

#include "FibonacciSequenceIterator.h"
#include <iostream>

// iterator constructor
FibonacciSequenceIterator::FibonacciSequenceIterator(const FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept
    : fSequenceObject(aSequenceObject), fIndex(aStart)
{
    if (aStart == 1)
    { //index 1 = fCurrent = 1
        fSequenceObject.reset();
    }
}

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

// prefix, next Fibonacci number
FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept
{
    //if has next number -> move on
    if (fSequenceObject.hasNext())
    {
        fSequenceObject.next();
        ++fIndex;
    }
    else
    {
        // Set fIndex out-of-scope to signal the end -> break the loop
        fIndex = MAX_FIBONACCI + 1;
    }
    return *this;
}

// postfix (extra unused argument)
FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept
{
    FibonacciSequenceIterator temp = *this;
    ++(*this);
    return temp;
}

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

bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator& aOther) const noexcept
```

```
53  {
54      return !(*this == aOther);
55  }
56
57  // return new iterator positioned at start
58  FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept {
59      return FibonacciSequenceIterator(fSequenceObject, 1);
60  }
61
62  // return new iterator positioned at limit
63  FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
64      return FibonacciSequenceIterator(fSequenceObject, MAX_FIBONACCI + 1);
65      //Indicate the end based on limits
66  }
67
```