

**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:** Luan Nguyen **Your student id:** 103812143

---

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: \_\_\_\_\_

```

//
// FibonacciSequenceGenerator.cpp
// Assignment2
//
// Created by Luan Nguyen on 15/4/2024.
//

#include <stdio.h>
#include "FibonacciSequenceGenerator.h"
#include <string>
#include <cassert>
#include <cstdint>

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;
}

// Converse type to bool
FibonacciSequenceGenerator::operator bool() const noexcept{
    return hasNext();
}

void FibonacciSequenceGenerator::reset() noexcept{
    fPrevious = 0;
    fCurrent = 1;
} // Reset the prev and current values

bool FibonacciSequenceGenerator::hasNext() const noexcept{
    long long lnext = fPrevious + fCurrent;
    return lnext>=0; // Return True if the next value is positive
}

void FibonacciSequenceGenerator::next() noexcept {
    // Perform the Fibonacci sequence calculation
    long long lnext = fPrevious + fCurrent;

    // Precondition assertion to guarantee no negative values
    assert(lnext >= 0);

    // Update previous and current values
    fPrevious = fCurrent;
    fCurrent = lnext;
}

```

```

//
// FibonacciSequenceIterator.cpp
// Assignment2
//
// Created by Luan Nguyen on 17/4/2024.
//

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

#include <cassert>

FibonacciSequenceIterator::FibonacciSequenceIterator(const
    FibonacciSequenceGenerator& aSequenceObject, long long aStart)
noexcept:
    fSequenceObject(aSequenceObject),
    fIndex(aStart)
{
    // assert(fSequenceObject);
}

// iterator
// Dereference operator
const long long& FibonacciSequenceIterator::operator*() const noexcept
{
    return *fSequenceObject;
}

FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept
{
    if (fSequenceObject.hasNext())
    {
        fSequenceObject.next();
    }
    ++fIndex;
    return *this;
}

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

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

```

```

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

FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept
{
    FibonacciSequenceGenerator lSequence =
        FibonacciSequenceGenerator(fSequenceObject.id());

    return FibonacciSequenceIterator(lSequence);
}

FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept
{
    FibonacciSequenceGenerator lSequence =
        FibonacciSequenceGenerator(fSequenceObject.id());
    long long lIndex = 1;
    while (lSequence.hasNext() == true)
    {
        lSequence.next();
        ++lIndex;
    }
    return FibonacciSequenceIterator(lSequence, lIndex+1);
}

```