COS30008 Semester August 2024 Ms. Siti Hawa

# Swinburne University of Technology

School of Science, Computing and Engineering Technologies

# ASSIGNMENT COVER SHEET

**Subject Code:** COS30008

**Subject Title:** Data Structures and Patterns

**Assignment number and title:** 2, Iterators

**Due date:** Wednesday, 9th October 2024, 23:59

**Lecturer:** Ms. Siti Hawa

## Your name: Amani Kamaruddin Bin Mikhail Raj Your student ID: J21035623

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 10/11/2024

Signature of Convener: ``



Problem Set 2: Iterators

Problem 1:

FibonacciSequenceGenerator.cpp

|  |
| --- |
| #include "FibonacciSequenceGenerator.h"  #include <cassert>  #include <limits>  #include <iostream>  #include <string>  #include <cstdint>  using namespace std;  FibonacciSequenceGenerator::FibonacciSequenceGenerator(const string &aID) noexcept      : fID(aID), fPrevious(0), fCurrent(1)  {  }  const string &FibonacciSequenceGenerator::id() const noexcept  {      return fID;  }  const long long &FibonacciSequenceGenerator::operator\*() const noexcept  {      // if(fCurrent < 0)      //     {      //         return fCurrent;      //     }      return fCurrent;  }  FibonacciSequenceGenerator::operator bool() const noexcept  {      // Checking if the current number is greater than 0      // return fCurrent > 0 && fCurrent <= numeric\_limits<int64\_t>::max();      // WHAT IS THE DIFFERENCE BETWEEN THIS AND THE ONE ABOVE      return hasNext();  }  void FibonacciSequenceGenerator::reset() noexcept  {      fPrevious = 0;      fCurrent = 1;  }  bool FibonacciSequenceGenerator::hasNext() const noexcept  {      // Check for potential overflow before it happens      if (fCurrent > numeric\_limits<int64\_t>::max() - fPrevious)      {          return false;      }      return true;  }  void FibonacciSequenceGenerator::next() noexcept  {      // I HATE THIS FUNCTION      if(!hasNext())      {          return;      }      assert(hasNext());      // long long temp = fCurrent;      // fCurrent += fPrevious;      // fPrevious = temp;      long long tempNext = fCurrent + fPrevious;      fPrevious = fCurrent;      fCurrent = tempNext;  } |

The Following Output:

|  |
| --- |
|  |

Problem 2:

|  |
| --- |
| #include "FibonacciSequenceIterator.h"  #include <cassert>  #include <limits>  #include <iostream>  #include <string>  #include <cstdint>  using namespace std;  FibonacciSequenceIterator::FibonacciSequenceIterator(const FibonacciSequenceGenerator &aSequenceObject, long long aStart) noexcept      : fSequenceObject(aSequenceObject), fIndex(aStart)  {  }  const long long &FibonacciSequenceIterator::operator\*() const noexcept  {      // return fSequenceObject.id();      // return fSequenceObject.current();      // return fIndex.current();      // this is the correct also, got it from following previous code in lab 6      // return \*fSequenceObject;      // idk what why and how does this work      return fSequenceObject.operator\*();  }  FibonacciSequenceIterator&FibonacciSequenceIterator::operator++() noexcept  {      // Check if there is a next Fibonacci number      // if (!fSequenceObject.hasNext())      // {      //     ++fIndex;// fIndex = 93;  // Set the index to a value beyond the 92nd Fibonacci so that it somehow stops the ieeration      //     return \*this; // Do not increment further      // }      // Otherwise, move to the next Fibonacci number      fSequenceObject.next();      ++fIndex;      return \*this;  }  FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept  {      // postfix      FibonacciSequenceIterator tempPrefix = \*this;      (\*this)++;      return tempPrefix;      // fSequenceObject.next();      // int64\_t tempPre = fIndex;      // fIndex++;      // return \*this;  }  bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator &aOther) const noexcept  {      return fIndex == aOther.fIndex;  }  bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator &aOther) const noexcept  {      return fIndex != aOther.fIndex;  }  FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept  {      // return FibonacciSequenceIterator(fSequenceObject, 0);      return FibonacciSequenceIterator(fSequenceObject, 1);  }  FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept  {      return FibonacciSequenceIterator(fSequenceObject,93);  } |

Output:

|  |
| --- |
|  |