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

```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
    assert(hasNext());

    // long long temp = fCurrent;
    // fCurrent += fPrevious;
    // fPrevious = temp;
    long long tempNext = fCurrent + fPrevious;
    fPrevious = fCurrent;
    fCurrent = tempNext;
}
```

The Following Output:

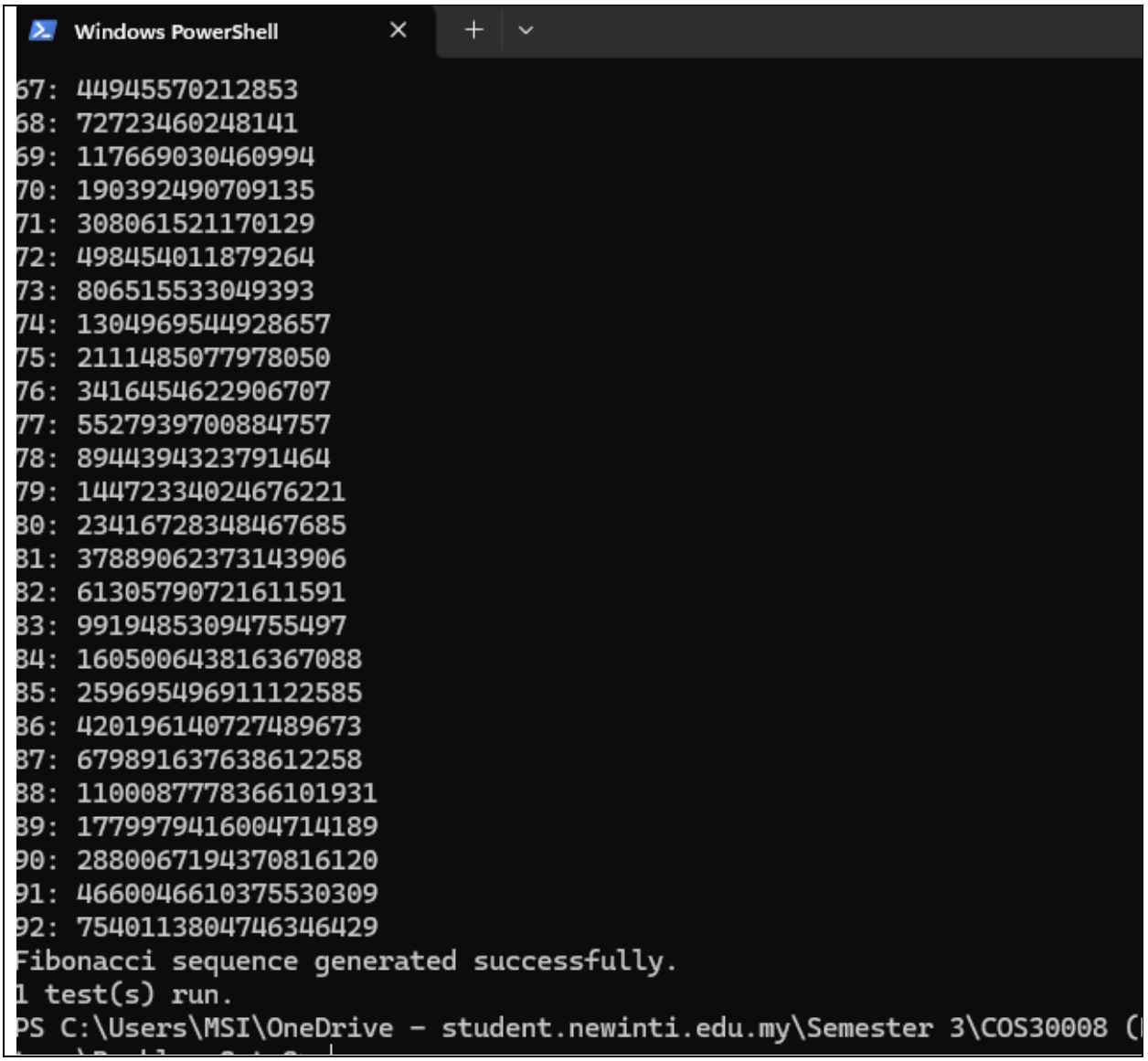
```
46: 1836311903
47: 2971215073
48: 4807526976
49: 7778742049
50: 12586269025
51: 20365011074
52: 32951280099
53: 53316291173
54: 86267571272
55: 139583862445
56: 225851433717
57: 365435296162
58: 591286729879
59: 956722026041
60: 1548008755920
61: 2504730781961
62: 4052739537881
63: 6557470319842
64: 10610209857723
65: 17167680177565
66: 27777890035288
67: 44945570212853
68: 72723460248141
69: 117669030460994
70: 190392490709135
71: 308061521170129
72: 498454011879264
73: 806515533049393
74: 1304969544928657
75: 2111485077978050
76: 3416454622906707
77: 5527939700884757
78: 8944394323791464
79: 14472334024676221
80: 23416728348467685
81: 37889062373143906
82: 61305790721611591
83: 99194853094755497
84: 160500643816367088
85: 259695496911122585
86: 420196140727489673
87: 679891637638612258
88: 1100087778366101931
89: 1779979416004714189
90: 2880067194370816120
91: 4660046610375530309
92: 7540113804746346429
Fibonacci sequence generated successfully.
1 test(s) run.
PS C:\Users\MSI\OneDrive - student.newinti.edu.my\Semester 3\COS30008 (DATA STRUCTURES AND PATTERNS)\C
OS30008-Data-structure\Problem-Set-2>
```

Problem 2:

```cpp
#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

    ++fIndex;
    fSequenceObject.next();
    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:


Windows PowerShell
67: 44945570212853
68: 72723460248141
69: 117669030460994
70: 190392490709135
71: 308061521170129
72: 498454011879264
73: 806515533049393
74: 1304969544928657
75: 2111485077978050
76: 3416454622906707
77: 5527939700884757
78: 8944394323791464
79: 14472334024676221
80: 23416728348467685
81: 37889062373143906
82: 61305790721611591
83: 99194853094755497
84: 160500643816367088
85: 259695496911122585
86: 420196140727489673
87: 679891637638612258
88: 1100087778366101931
89: 1779979416004714189
90: 2880067194370816120
91: 4660046610375530309
92: 7540113804746346429
Fibonacci sequence generated successfully.
1 test(s) run.
PS C:\Users\MSI\OneDrive - student.newinti.edu.my\Semester 3\COS30008 (