# **Swinburne University of Technology**

Faculty of Science, Engineering and Technology

## **ASSIGNMENT COVER SHEET**

Subject Code: Subject Title: Assignment number and title Due date: Lecturer:	COS30008  Data Structures & Pate 2 - Iterators  Monday, 22 April, 202  Dr. Markus Lumpe	
Your name: Avery Flannery	Your	student id: 104416957
Marker's comments:		
Problem	Marks	Obtained
1	40	
2	70	
Total	110	
Extension certification:  This assignment has been given	an extension and is now d	ue on
Signature of Convener:		

#### Problem 1 – FibonacciSequenceGenerator.cpp

```
//COS30008 - 104416957 - Avery Flannery. Semester 1 2024
//Fibonacci Sequence Generator
#include "FibonacciSequenceGenerator.h"
#include <cassert>
// constructor
FibonacciSequenceGenerator::FibonacciSequenceGenerator(const std::string& aID)
    fID(aID), fPrevious(0), fCurrent(1) {}
// get sequence id
const std::string& FibonacciSequenceGenerator::id() const noexcept {
   return fID;
}
//deference operator to get current Fibonacci number
const long long& FibonacciSequenceGenerator::operator*() const noexcept {
   return fCurrent; // returning the current number
}
// converting operator to bool
FibonacciSequenceGenerator::operator bool() const noexcept {
   return hasNext(); // check if there is a next Fibonacci number
}
// used to reset the sequence generator
void FibonacciSequenceGenerator::reset() noexcept {
    fPrevious = 0; // return previous
   fCurrent = 1; // return current
}
// used to check if there is a next Fibonacci number
bool FibonacciSequenceGenerator::hasNext() const noexcept {
    long long next = fPrevious + fCurrent; // calculates the next number
   return next >= 0; // checks if the number is negative
}
// used to go on to the next Fibonacci number in the sequence
void FibonacciSequenceGenerator::next() noexcept {
   long long next = fPrevious + fCurrent; // calculates the number
   fPrevious = fCurrent; // updates the previous Fibonacci number
   fCurrent = next; // updates the current Fibonacci number
}
```

#### Problem 2 – FibonacciSequenceIterator.cpp

```
//COS30008 - 104416957 - Avery Flannery. Semester 1 2024
//Fibonacci Sequence Iterator
#include "FibonacciSequenceIterator.h"
// constructor
FibonacciSequenceIterator::FibonacciSequenceIterator(const
FibonacciSequenceGenerator& aSequenceObject, long long aStart) noexcept
    : fSequenceObject(aSequenceObject), fIndex(aStart) {}
// deference operator
const long long& FibonacciSequenceIterator::operator*() const noexcept {
   return *fSequenceObject; // deferring the object
}
// prefix increment operator
FibonacciSequenceIterator& FibonacciSequenceIterator::operator++() noexcept {
    ++fIndex; // increments iterator position
    fSequenceObject.next(); //move to the next Fibonacci number
   return *this;
}
// postfix increment operator
FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int) noexcept {
    FibonacciSequenceIterator temp = *this; // create a copy of the current
iterator
    ++(*this); // increment the iterator position
   return temp; // return the copy
}
// == equals comparison operator
bool FibonacciSequenceIterator::operator==(const FibonacciSequenceIterator&
aOther) const noexcept {
    // comparing the sequence objects' ID and iterator positions
   return (fSequenceObject.id() == aOther.fSequenceObject.id()) && (fIndex ==
aOther.fIndex);
// != does not equal comparison operator
bool FibonacciSequenceIterator::operator!=(const FibonacciSequenceIterator&
aOther) const noexcept {
   return !(*this == a0ther);
FibonacciSequenceIterator FibonacciSequenceIterator::begin() const noexcept {
    FibonacciSequenceIterator beginIterator = *this;
   beginIterator.fIndex = 1; // sets the iterator to 1 to start
   beginIterator.fSequenceObject.reset();
   return beginIterator; // returns the iterator
}
FibonacciSequenceIterator FibonacciSequenceIterator::end() const noexcept {
    FibonacciSequenceIterator endIterator = *this;
    endIterator.fIndex = 93; // sets the iterator position to 93 at the end
   return endIterator; // returns the iterator
}
```

### **Output**

```
Microsoft Visual Studio Debug Console
ibonacci sequence P1 for long long:
            1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4481
6765
10946
17711
28657
46368
75213
317811
514229
             2178309
3524578
5702887
9227465
14930352
             24157817
39088169
63245986
102334155
165580141
             267914296
433494437
701408733
1134903170
1836311903
            1836311903
2971215073
4807526976
7778742049
12586269025
20365011074
32951280099
53316291173
              86267571272
139583862445
          8026/37/12/2

139583862445

225851433717

365435296162

591286729879

956722026641

1548068755920

2504730781961

4852739537881

6557470319842

10616209857723

17167680177565

27777890035288

44945570212853

72723460248141

11766993466994

190392490709135

3688661521170129

488454611879264

806515533049393
            3686013211/6129
498454611879264
866515533649393
1364969544928657
2111485677978850
3416454622906707
5527939708884757
8944394323791464
14472334024676221
23416728348467685
37889662373143906
61305790721611591
99194853094755497
1605696438161367088
259055496911122885
420196140727489673
679891637638612258
1100087778366101931
1779979416004714189
2880667194370816120
4660046619375539309
7540113804746346429
              7540113804746346429
onacci sequence generated successfully.
```

```
ibonacci sequence P2 for long long:
 2: 1
3: 2
4: 3
5: 5
6: 8
7: 13
8: 21
9: 34
10: 55
11: 89
12: 144
13: 233

14: 377

15: 610

16: 987

17: 1597

18: 2584

19: 4181

20: 6765

21: 18946

22: 17711

23: 28657

24: 46368

25: 75025

26: 121393

27: 196418

28: 317811

29: 514229

30: 832040

31: 1346269

32: 2178309

33: 3524578

34: 5702887
  35: 9227465
36: 14930352
  36: 14936352
37: 24157817
38: 39688169
39: 63245986
40: 102334155
41: 165580141
 41: 16558141
42: 267914296
43: 433494437
44: 701408733
45: 1134903170
46: 1836311903
  47: 2971215073
48: 4807526976
49: 7778742049
50: 12586269025
51: 20365269025
  51: 20365011074

52: 32951280099

53: 53316291173

54: 86267571272

55: 139583862445

56: 225851433717
  57: 365435296162
58: 591286729879
    59: 956722026041
60: 1548008755920
61: 2504730781961
 61: 2504730781961
62: 4652739537881
63: 6557470319842
64: 10610209857723
65: 17167680177565
66: 27777890035288
67: 44945570212853
68: 72723460248141
69: 117669030460994
70: 190392490709135
71: 308061521170129
72: 498454011879264
  72: 498454011879264
73: 806515533049393
  73: 886515534649393
74: 1304969544928657
75: 2111485077978050
76: 3416454622906707
77: 5527939700884757
78: 8944394323791464
78: 8944394323791464
79: 14472334624676221
80: 23416728348467685
81: 37889662373143966
82: 61305790721611591
83: 99194853094755497
84: 160509643816367088
85: 259695496911122585
 85: 259695496911122585
86: 420196140727489673
87: 679891637638612258
88: 1100087778366101931
89: 1779979416004714189
90: 2880067194370816120
91: 4660046610375530309
92: 7540113804746346429
     Fibonacci sequence generated successfully.
2 test(s) run.
```