FibonacciSequence.cpp

```cpp
#include "FibonacciSequence.h"
#include "FibonacciSequenceIterator.h"
#include <stdexcept>


FibonacciSequence::FibonacciSequence(unsigned long aLimit)
{
    fLimit = aLimit;
    fPrevious = 0;
    fCurrent = 1;
    fPosition = 1;
}


const unsigned long& FibonacciSequence::current() const
{
    return fCurrent;
}


void FibonacciSequence::advance()
{
    unsigned long next = fCurrent + fPrevious;
    fPrevious = fCurrent;
    fCurrent = next;
    if (fLimit < fPosition)
    {
        throw std::out_of_range("Limit exceeded, please try again");
    }
    fPosition++;
}


const unsigned long& FibonacciSequence::getLimit() const
{
    return fLimit;
}


FibonacciSequenceIterator FibonacciSequence::begin()
```

```
{
   *this = FibonacciSequence(getLimit());
   return FibonacciSequenceIterator(*this, 1);
}


FibonacciSequenceIterator FibonacciSequence::end()
{
   return FibonacciSequenceIterator(*this, getLimit()+1);
```

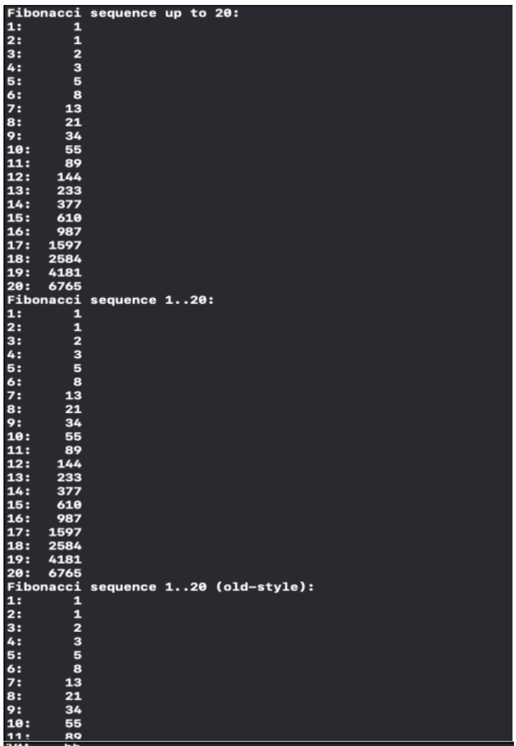
FibonacciSequenceIterator.cpp

```
#include "FibonacciSequenceIterator.h"
#include "FibonacciSequence.h"


FibonacciSequenceIterator::FibonacciSequenceIterator(FibonacciSequence& aSequenceObject, unsigned long
aStart) : fSequenceObject(aSequenceObject), fIndex(aStart)
{
}


const unsigned long& FibonacciSequenceIterator::operator*() const
{
   return fSequenceObject.current();
}


FibonacciSequenceIterator& FibonacciSequenceIterator::operator++()
{
   fSequenceObject.advance();
   fIndex++;
   return *this;
}


FibonacciSequenceIterator FibonacciSequenceIterator::operator++(int)
{
     FibonacciSequenceIterator temp = *this;
     fSequenceObject.advance();
     fIndex++;
```

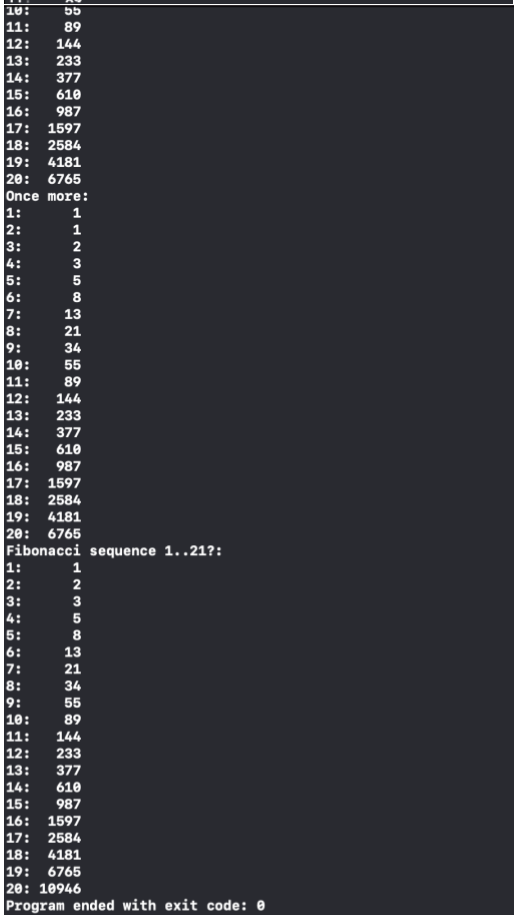
```
    return temp;
}


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


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


FibonacciSequenceIterator FibonacciSequenceIterator::begin() const
{
  return fSequenceObject.begin();
}


FibonacciSequenceIterator FibonacciSequenceIterator::end() const
{
  return fSequenceObject.end();
}
```

Problem 3:
The last output was off by 1 Fibonacci number because in the while loop it implements *lIteratorC++. That skips the first IteratorC.

```
while ( lIteratorC != lIteratorC.end() )
{
    cout << c++ << ":\t" << setw(5) << *lIteratorC++ << endl;
}
```



```
Fibonacci sequence up to 20:
1:      1
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
Fibonacci sequence 1..20:
1:      1
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
Fibonacci sequence 1..20 (old-style):
1:      1
2:      1
3:      2
4:      3
5:      5
6:      8
7:     13
8:     21
9:     34
10:    55
```



```
10:    55
11:    89
12:   144
13:   233
14:   377
15:   610
16:   987
17:  1597
18:  2584
19:  4181
20:  6765
Once more:
1:      1
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
Fibonacci sequence 1..21?:
1:      1
2:      2
3:      3
4:      5
5:      8
6:     13
7:     21
8:     34
9:     55
10:    89
11:   144
12:   233
13:   377
14:   610
15:   987
16:  1597
17:  2584
18:  4181
19:  6765
20: 10946
Program ended with exit code: 0
```