Swinburne University of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Assignment number and title: 2, Indexers, Iterators, and Inheritance
Due date: April 20, 2021, 16:00
Lecturer: Dr. Markus Lumpe

Your name: Sam Cross **Your student id:** 5840422

Check Tutorial	Wed 08:30	Wed 10:30	Wed 16:30	Thurs 08:30	Thurs 10:30	Thurs 14:30	Thurs 16:30	Fri 08:30	Fri 10:30	Fri 14:30
				✓						

Marker's comments:

Problem	Marks	Obtained
1a	54	
1b	64	
2	44	
Total	162	

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener: _____

```cpp
#include "IntVector.h"
#include <cstddef>
#include "IntSorter.h"
#include "IntVectorIterator.h"
#include <stdexcept>

using namespace std;

IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements)
{
    fNumberOfElements = aNumberOfElements;
    fElements = new int[fNumberOfElements];

    for (size_t i = 0; i < fNumberOfElements; i++)
    {
        fElements[i] = aArrayOfIntegers[i];
    }
}

IntVector::~IntVector()
{
    delete[] fElements;
}

size_t IntVector::size() const
{
    return fNumberOfElements;
}

void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex)
{
    if (aSourceIndex < fNumberOfElements && aTargetIndex < fNumberOfElements)
    {
        size_t tempIndex = fElements[aSourceIndex];
        fElements[aSourceIndex] = fElements[aTargetIndex];
        fElements[aTargetIndex] = tempIndex;

        return;
    }
    throw
        out_of_range("Invalid index(es).");
}

void IntVector::sort(const IntSorter& aSorter)
{
    aSorter(*this);
}

const int IntVector::operator[](size_t aIndex) const
{
    if (aIndex < fNumberOfElements)
    {
        return fElements[aIndex];
```

```cpp
#include "IntVectorIterator.h"
#include "IntVector.h"

#include <iostream>

using namespace std;

IntVectorIterator::IntVectorIterator(const IntVector& aContainer, size_t aStart) : fContainer(aContainer),  fPosition(aStart)
{

}

const int IntVectorIterator::operator*() const
{
    return fContainer[fPosition];
}

IntVectorIterator& IntVectorIterator::operator++()
{
    fPosition++;
    return *this;
}

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

bool IntVectorIterator::operator==(const IntVectorIterator& aRHS) const
{
    return
        &fContainer == &aRHS.fContainer &&
        fPosition == aRHS.fPosition;
}
bool IntVectorIterator::operator!=(const IntVectorIterator& aRHS) const
{
    return !(*this == aRHS);
}

IntVectorIterator IntVectorIterator::begin() const
{
    IntVectorIterator iter = *this;
    iter.fPosition = 0;
    iter.operator++();
    return iter;
}

IntVectorIterator IntVectorIterator::end() const
{
    IntVectorIterator iter = *this;
```

```cpp
#include "CocktailShakerSort.h"
#include "IntVector.h"

using namespace std;

void CocktailShakerSort::operator()(IntVector& aContainer) const
{
    int beginIndex = 0;
    int endIndex = aContainer.size() - 1;

    while (beginIndex < endIndex)
    {
        for (int i = beginIndex; i < endIndex; i++)
        {
            int a = aContainer[i];
            int b = aContainer[i + 1];

            if (aContainer[i] > aContainer[i + 1])
            {
                aContainer.swap(i, i + 1);
            }
        }

        endIndex -= 1;

        for (int i = endIndex; i > beginIndex; i--)
        {
            int a = aContainer[i];
            int b = aContainer[i - 1];

            if (aContainer[i] < aContainer[i - 1])
            {
                aContainer.swap(i, i - 1);
            }
        }

        beginIndex += 1;
    }
}
```