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, Method Overriding, and Lambdas
Due date: April 7, 2022, 14:30
Lecturer: Dr. Markus Lumpe

Your name:_____ **Your student id:**_____

Check Tutorial	Mon 10:30	Mon 14:30	Tues 08:30	Tues 10:30	Tues 12:30	Tues 14:30	Tues 16:30	Wed 08:30	Wed 10:30	Wed 12:30	Wed 14:30

Marker's comments:

Problem	Marks	Obtained
1	48	
2	30+10= 40	
3	58	
Total	146	

Extension certification:

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

Signature of Convener:_____

```cpp
#include "IntVector.h"
#include <cstddef>
#include <stdexcept>
// #include <type_traits>
#include <utility>
// #include <vcruntime_new.h>

IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements) {
    this->fNumberOfElements = aNumberOfElements;
    this->fElements = new int[aNumberOfElements];
    for (int i = 0; i < aNumberOfElements; i++) {
        this->fElements[i] = aArrayOfIntegers[i];
    }
}

IntVector::~IntVector() {
    // delete this->fNumberOfElements;
    delete[] this->fElements;
}

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

const int IntVector::operator[](size_t aIndex) const {
    // we don't need the *this here, or am I missing something?
    if (aIndex < 0 || aIndex >= this->fNumberOfElements) {
        throw std::out_of_range("Illegal vector index");
    }
    return this->fElements[aIndex];
}

const int IntVector::get(size_t aIndex) const {
    // I guess this is where it actually goes?
    return (*this)[aIndex];
}

void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex) {
    if (aSourceIndex < 0 || aSourceIndex >= this->fNumberOfElements ||
        aTargetIndex < 0 || aTargetIndex >= this->fNumberOfElements) {
        throw std::out_of_range("Illegal vector indicies");
    }

    // yes this function actually exists
    std::swap(this->fElements[aSourceIndex], this->fElements[aTargetIndex]);
}
```

```cpp
#include "ShakerSortableIntVector.h"
#include "SortableIntVector.h"
#include <cstddef>
ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers []),
                                                 size_t aNumberOfElements)
    : SortableIntVector(aArrayOfIntegers, aNumberOfElements) {}

void ShakerSortableIntVector::sort(Comparable aOrderFunction) {
    size_t n = this->size();
    size_t beginIndex = 0, endIndex = n - 1;
    while (beginIndex < endIndex) {
        for (size_t i = beginIndex; i <= endIndex - 1; i++) {
            // if a[i] > a[i + 1]

            // aOrderFunction means a <= b => inverse of that is a > b
            if (aOrderFunction(this->get(i), this->get(i + 1))) {
                this->swap(i, i + 1);
            }
        }

        endIndex -= 1;

        for (size_t i = endIndex; i >= beginIndex + 1; i--) {
            if (!aOrderFunction(this->get(i), this->get(i - 1)) &&
                this->get(i) != this->get(i - 1)) {

                this->swap(i, i - 1);
            }
        }

        beginIndex += 1;
    }
}
```

```cpp
#include "SortableIntVector.h"
#include "IntVector.h"
#include <cstddef>

SortableIntVector::SortableIntVector(const int aArrayOfIntegers[],
                                     size_t aNumberOfElements)
    : IntVector(aArrayOfIntegers, aNumberOfElements) {}

void SortableIntVector::sort(Comparable aOrderFunction) {
    size_t n = this->size();
    for (size_t i = 0; i <= n - 1; i++) {
        for (size_t j = n - 1; j >= i + 1; j--) {

            // turns out you can just do this. If you flip it, it will sort in
            // reverse and the reason is that we are following the sort lambda
            // in Main. so in essence, you could define aOrderFunction as the
            // lambda to check a > b, but that would go against the definition
            // of our Comparable here, so I won't do that.

            if (aOrderFunction(this->get(j), this->get(j - 1))) {
                // swap(a[j], a[j - 1])
                this->swap(j, j - 1);
            }
        }
    }
}
```