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: Nguyen Duy Anh Tu **Your student id:** 104188405

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:____________________________

```
#include "IntVector.h"
#include "stdexcept"

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;
}

const int IntVector::get(size_t aIndex) const
{
    return (*this)[aIndex];
}

void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex)
{
    if (aSourceIndex >= fNumberOfElements || aTargetIndex >= fNumberOfElements)
throw std::out_of_range("Illegal vector indices");
    int temp = fElements[aSourceIndex];
    fElements[aSourceIndex] = fElements[aTargetIndex];
    fElements[aTargetIndex] = temp;
}

const int IntVector::operator[](size_t aIndex) const
{
    if (aIndex >= fNumberOfElements) throw std::out_of_range("Illegal vector
index");
    return fElements[aIndex];
}
```

```
#include "SortableIntVector.h"

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

void SortableIntVector::sort(Comparable aOrderFunction)
{
	for (size_t i = 0; i < (*this).size(); i++)
	{
		for (size_t j = (*this).size() - 1; j > i; j--)
		{
			if (aOrderFunction(get(j - 1), get(j)))
			{
				(*this).swap(j, j - 1);
			}
		}
	}
}
```

```
#include "ShakerSortableIntVector.h"

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

void ShakerSortableIntVector::sort(Comparable aOrderFunction)
{
	for (size_t i = 0; i < (*this).size(); i++)
	{
		for (size_t j = (*this).size() - 1; j > i; j--)
		{
			if (aOrderFunction(get(j - 1), get(j)))
			{
				(*this).swap(j, j - 1);
			}
		}
	}
 }
```