**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:** Dao Khanh Nga Thi **Your student id:** 104177393

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | X | | | | | | | |

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:____________________________

**Problem 1:** IntVector.cpp

```cpp
#include "IntVector.h"
#include <stdexcept>

IntVector::IntVector(const int* aArrayOfIntegers, size_t aNumberOfElements)
    : fNumberOfElements(aNumberOfElements)
    , fElements(new int[aNumberOfElements])
{
    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
{
    if (aIndex >= fNumberOfElements)
    {
        throw std::out_of_range("Illegal vector index");
    }

    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];
}
```

**Problem 2:** SortableIntVector.cpp

```
#include "SortableIntVector.h"
#include <functional>

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

void SortableIntVector::sort(std::function<bool(int, int)> aOrderFunction)
{
    for (size_t i = 0; i < size(); ++i)
    {
        for (size_t j = size() - 1; j > i; --j)
        {
            if (aOrderFunction((*this)[j - 1], (*this)[j]))
            {
                swap(j - 1, j);
            }
        }
    }
}
```

*Main_PS2.cpp

```
lVector.sort([](int aLeft, int aRight) -> bool { return aLeft <= aRight; });
```

**Problem 3:** ShakerSortableIntVector.cpp

```
#include "ShakerSortableIntVector.h"

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

void ShakerSortableIntVector::sort(Comparable aOrderFunction)
{
    size_t left = 0;
    size_t right = (*this).size() - 1;

    while (left < right)
    {
        for (size_t i = left; i < right; i++)
        {
            if (!aOrderFunction(get(i + 1), get(i)))
            {
                (*this).swap(i, i + 1);
            }
        }

        right--;

        for (size_t i = right; i > left; i--)
        {
            if (!aOrderFunction(get(i), get(i - 1)))
            {
                (*this).swap(i - 1, i);
            }
        }

        left++;
    }
}
```