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 Duc Chung **Your student id:** 104972970

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:_____

Problem 1:

Invector.cpp:

```
#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 index");
   }

   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"

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

void SortableIntVector::sort(Comparable aOrderFunction)
{

    for (size_t i = 0; i < size(); i++)
    {
        for (size_t j = 0; j < size() - i - 1; j++)
        {
            int leftElement = get(j);
            int rightElement = get(j + 1);

            bool inCorrectOrder = aOrderFunction(leftElement, rightElement);

            if (!inCorrectOrder)
            {
                swap(j, j + 1);
            }
        }
    }
}
```

Main_PS2.cpp:

```
#ifdef P2
#include "SortableIntVector.h"
void runP2()
{
int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
size_t lArrayLength = sizeof(lArray) / sizeof(int);
SortableIntVector lVector(lArray, lArrayLength);
cout << "Bubble Sort:" << endl;
cout << "Before sorting:" << endl;
for (size_t i = 0; i < lVector.size(); i++)

{
cout << lVector[i] << ' ';
}
cout << endl;

lVector.sort([](int left, int right) { return left < right; });

cout << "After sorting:" << endl;
for (size_t i = 0; i < lVector.size(); i++)
{
```

```
cout << lVector[i] << ' ';
}
cout << endl;
}
#endif
```

Problem 3:

ShakerSortableIntVector.cpp:

```
#include "ShakerSortableIntVector.h"
ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers[], size_t
aNumberOfElements)
   : SortableIntVector(aArrayOfIntegers, aNumberOfElements) {}
void ShakerSortableIntVector::sort(Comparable aOrderFunction)
{
   bool swapped = true;
   size_t start = 0;
   size_t end = size() - 1;

   while (swapped)
   {
      swapped = false;
      for (size_t i = start; i < end; ++i)
      {
         if (!aOrderFunction(get(i), get(i + 1)))
         {
            swap(i, i + 1);
            swapped = true;
         }
      }

      if (!swapped) break;

      swapped = false;

      --end;

      for (size_t i = end; i > start; --i)
      {
         if (!aOrderFunction(get(i - 1), get(i)))
         {
            swap(i - 1, i);
            swapped = true;
         }
      }

      ++start;
   }
}
```