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:	Sunday, October 22, 2023, 23:59 (VN Time)
Lecturer:	Dr. Van Dai PHAM

Your name: Nguyen Quoc Thang **Your student id:** 104193360

Check Tutorial	Mon 10:30	Thursday 10:00 Innovation Lab	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<iostream>
#include<vector>
#include<stdexcept>
using namespace std;
#include"IntVector.h"
IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements) {
    this->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 >= 0 && aTargetIndex < fNumberOfElements && aSourceIndex ↩
       < fNumberOfElements ) {
         size_t intermediateSource = fElements[aSourceIndex];
         fElements[aSourceIndex] = fElements[aTargetIndex];
         fElements[aTargetIndex] = intermediateSource;
     }
     else
     {
         throw out_of_range("Illegal vector indices");
     }
 }
 const int IntVector::operator[](size_t aIndex) const {
     if (aIndex >= fNumberOfElements) {
         throw out_of_range("Illegal vector index");
     }
     else {
        return fElements[aIndex];
     }

 }
```

```
#include<iostream>
#include"SortableIntVector.h"
using namespace std;
SortableIntVector::SortableIntVector(const int aArrayOfIntegers[], size_t     ↩
  aNumberOfElements) : IntVector( aArrayOfIntegers,  aNumberOfElements){

}
 void SortableIntVector::sort(Comparable aOrderFunction) {

     for (size_t i = 0; i < size() - 1; i++) {
         for (size_t j = 0; j < size() - i - 1 ; j++) {
             if (aOrderFunction(get(j) ,get(j+1))) {
                 swap(j,j+1);
             }
         }
     }
}
```

```
#include<iostream>
#include"ShakerSortableIntVector.h"
using namespace std;
ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers[], 
  size_t aNumberOfElements) : SortableIntVector(aArrayOfIntegers , 
  aNumberOfElements){

}
void ShakerSortableIntVector::sort(Comparable aOrderFunction ) {
    for (size_t i = 0; i < size() / 2; i++) {
        for (size_t j = 0; j < size() - i - 1; j++) {
            if (aOrderFunction(get(j), get(j + 1))) {
                swap(j, j + 1);
            }
        }
        for (size_t j = size() - 1 - i; j > i; j--) {
            if (aOrderFunction(get(j - 1), get(j))) {
                swap(j - 1 , j );
            }
        }
    }
}
```

```cpp
// Problem Set 2, 2022

#include <iostream>
#include <stdexcept>

using namespace std;

//#define P1
//#define P2
#define P3

#ifdef P1

#include "IntVector.h"

void runP1()
{
    int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
    size_t lArrayLength = sizeof(lArray) / sizeof(int);

    IntVector lVector(lArray, lArrayLength);

    cout << "Test range check:" << endl;

    try
    {
        int lValue = lVector[lArrayLength];

        cerr << "Error, you should not see " << lValue << " here!" << endl;
    }
    catch (out_of_range e)
    {
        cerr << "Properly caught error: " << e.what() << endl;
    }
    catch (...)
    {
        cerr << "This message must not be printed!" << endl;
    }

    cout << "Test swap:" << endl;

    try
    {
        cout << "lVector[3] = " << lVector[3] << endl;
        cout << "lVector[6] = " << lVector[6] << endl;

        lVector.swap(3, 6);
        cout << "lVector.get( 3 ) = " << lVector.get(3) << endl;
        cout << "lVector.get( 6 ) = " << lVector.get(6) << endl;

        lVector.swap(5, 20);

        cerr << "Error, you should not see this message!" << endl;
    }
    catch (out_of_range e)
    {
        cerr << "Properly caught error: " << e.what() << endl;
    }
    catch (...)
    {
        cerr << "Error, this message must not be printed!" << endl;
    }
}

#endif

#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 a, int b) -> bool {return a >= b;});

    cout << "After sorting:" << endl;

    for (size_t i = 0; i < lVector.size(); i++)
    {
        cout << lVector[i] << ' ';
    }

    cout << endl;
}

#endif

#ifdef P3

#include "ShakerSortableIntVector.h"

void runP3()
{
    int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
    size_t lArrayLength = sizeof(lArray) / sizeof(int);

    ShakerSortableIntVector lVector(lArray, lArrayLength);

    cout << "Cocktail Shaker Sort:" << endl;

    cout << "Before sorting:" << endl;

    for (size_t i = 0; i < lVector.size(); i++)
    {
        cout << lVector[i] << ' ';
    }

    cout << endl;

    // sort in decreasing order
    lVector.sort();

    cout << "After sorting:" << endl;

    for (size_t i = 0; i < lVector.size(); i++)
    {
        cout << lVector[i] << ' ';
    }

    cout << endl;
}

#endif

int main()
{
#ifdef P1

    runP1();
```

```
#endif

#ifdef P2

    runP2();

#endif

#ifdef P3

    runP3();

#endif

    return 0;
}
```