COS30008 Fall Semester, 2023 Dr. Markus Lumpe

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: GIA HUNG LE **Your student id:** 104057754

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 "IntVector.h"
#include <cstddef>
#include <algorithm>

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) {
		return;
	}
	std::swap(fElements[aSourceIndex], fElements[aTargetIndex]);
}

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

```
#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 << "Properly caught error: Illegal vector indices" << 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 = std::size(lArray);

    SortableIntVector lVector(lArray, lArrayLength);

    std::cout << "Bubble Sort:" << std::endl;

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

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

    std::cout << std::endl;
    lVector.sort([](int a, int b) { return a < b; });

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

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

    std::cout << std::endl;
}

#endif

#ifdef P3

#include "ShakerSortableIntVector.h"

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

    ShakerSortableIntVector lVector(lArray, lArrayLength);

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

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

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

    std::cout << std::endl;

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

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

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

    std::cout << std::endl;
}
#endif

int main()
{
#ifdef P1

    runP1();

#endif

#ifdef P2

    runP2();

#endif

#ifdef P3

    runP3();

#endif

    return 0;
}
```

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

void ShakerSortableIntVector::sort(Comparable aOrderFunction) {
    bool swapped = true;
    size_t n = size();
    size_t start = 0;
    size_t end = n - 1;

    while (swapped) {
        swapped = false;

        for (size_t i = start; i < end; ++i) {
            if (aOrderFunction((*this)[i], (*this)[i + 1])) {
                swap(i, i + 1);
                swapped = true;
            }
        }
        if (!swapped) break;

        --end;


        for (size_t i = end - 1; i >= start; --i) {
            if (aOrderFunction((*this)[i], (*this)[i + 1])) {
                swap(i, i + 1);
                swapped = true;
            }
        }


        ++start;
    }
      }
```

```
#include "SortableIntVector.h"

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

void SortableIntVector::sort(Comparable aOrderFunction) {
    bool swapped = true;
    size_t n = size();
    while (swapped) {
        swapped = false;
        for (size_t i = 1; i < n; ++i) {
            if (aOrderFunction((*this)[i], (*this)[i - 1])) {
                swap(i, i - 1);
                swapped = true;
            }
        }
        --n;
    }
}
```