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: Jamie Kozminska **Your student id:** 101114436

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:_____

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

using namespace std;

IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements)
{
    fNumberOfElements = aNumberOfElements;      // passing in the number of elements
    fElements = new int[fNumberOfElements];     //creating an integer array fElements

    for (size_t i = 0; i < fNumberOfElements; i++)
    {
        fElements[i] = aArrayOfIntegers[i];     //looping through and populating the array with values
    }
}

IntVector::~IntVector()
{
    delete[] fElements;         //destructor deletes the elements and array
}

size_t IntVector::size() const
{
    return fNumberOfElements;       //gives back the size of the vector
}

const int IntVector::get(size_t aIndex) const
{
        if (aIndex >= fNumberOfElements)
        {
            throw out_of_range("Illegal vector indices.");
        }
        return  (*this)[aIndex];
}

void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex) //the member function swap() takes two indices and, if they are within range, swaps the
                                                               //
                    corresponding array elements in an IntVector object. We need swap() for sorting.
{
    int temp = get(aSourceIndex);
    fElements[aSourceIndex] = get(aTargetIndex);
    fElements[aTargetIndex] = temp;


```

```
44      if (aTargetIndex > fNumberOfElements)
45      {
46          throw out_of_range("Out of range");
47      }
48  }
49
50  const int IntVector::operator[](size_t aIndex) const
51  {
52          if (aIndex >= 0 && aIndex < fNumberOfElements)
53          {
54              return fElements[aIndex];
55          }
56          throw out_of_range("illegal aIndex.");
57  }
58
```

```cpp

#include "SortableIntVector.h"

using namespace std;

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

void SortableIntVector::sort(Comparable aOrderFunction)
{
    for (size_t i = 0; i < size() - 1; i++)         //loop through size of the array -1
    {
        for (size_t j = 0; j < size() - i - 1; j++) //inner loop
        {
            if (aOrderFunction(get(j), get(j + 1)))     // if the comparable returns true
            {
                swap(j, j + 1);         //swap the values
            }
        }
    }
}
```

```cpp

#include "SortableIntVector.h"
#include "ShakerSortableIntVector.h"

using namespace std;



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


void ShakerSortableIntVector::sort(Comparable aOrderFunction)
{
    int start = 0;
    int end = size() - 1;
    while (start < end)
    {
        for (int i = start; i < end; ++i)
        {
            if (aOrderFunction(get(i), get(i + 1)))     // if the comparable returns true
            {
                swap(i, i + 1);         //swap the values
            }
        }
        --end;

        for (int i = end - 1; i >= start; --i)
        {
            if (aOrderFunction(get(i), get(i + 1)))     // if the comparable returns true
            {
                swap(i, i + 1);         //swap the values
            }
        }
        ++start;
    }
}


```

```cpp

// Problem Set 2, 2022

#include <iostream>
#include <stdexcept>
#include<array>
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 );

```

```
50          cout << "lVector.get( 3 ) = " << lVector.get( 3 ) << endl;
51          cout << "lVector.get( 6 ) = " << lVector.get( 6 ) << endl;
52
53          lVector.swap( 5, 20 );
54
55          cerr << "Error, you should not see this message!" << endl;
56      }
57      catch (out_of_range e)
58      {
59          cerr << "Properly caught error: " << e.what() << endl;
60      }
61      catch (...)
62      {
63          cerr << "Error, this message must not be printed!" << endl; //this is
              printing
64      }
65  }
66
67  #endif
68
69  #ifdef P2
70
71  #include "SortableIntVector.h"
72
73  void runP2()
74  {
75      int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
76      size_t lArrayLength = sizeof(lArray) / sizeof(int);
77
78      SortableIntVector lVector(lArray, lArrayLength);
79
80      cout << "Bubble Sort:" << endl;
81
82      cout << "Before sorting:" << endl;
83
84      for (size_t i = 0; i < lVector.size(); i++)
85      {
86          cout << lVector[i] << ' ';
87      }
88
89      cout << endl;
90
91
92      lVector.sort([](int aLeft, int aRight) { return aLeft >= aRight; });
93
94      cout << "After sorting:" << endl;
95
96      for ( size_t i = 0; i < lVector.size(); i++ )
97      {
```

```
 98          cout << lVector[i] << ' ';
 99      }
100
101      cout << endl;
102  }
103
104  #endif
105
106  #ifdef P3
107
108  #include "ShakerSortableIntVector.h"
109
110  void runP3()
111  {
112      int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
113      size_t lArrayLength = sizeof(lArray) / sizeof(int);
114
115      ShakerSortableIntVector lVector( lArray, lArrayLength );
116
117      cout << "Cocktail Shaker Sort:" << endl;
118
119      cout << "Before sorting:" << endl;
120
121      for ( size_t i = 0; i < lVector.size(); i++ )
122      {
123          cout << lVector[i] << ' ';
124      }
125
126      cout << endl;
127
128      // sort in decreasing order
129      lVector.sort();
130      cout << "After sorting:" << endl;
131
132      for ( size_t i = 0; i < lVector.size(); i++ )
133      {
134          cout << lVector[i] << ' ';
135      }
136
137      cout << endl;
138  }
139
140  #endif
141
142  int main()
143  {
144  #ifdef P1
145
146      runP1();
```

```
147
148 #endif
149
150 #ifdef P2
151
152     runP2();
153
154 #endif
155
156 #ifdef P3
157
158     runP3();
159
160 #endif
161
162     return 0;
163 }
164
```