

**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:** \_\_\_\_\_ **Your student id:** \_\_\_\_\_

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: \_\_\_\_\_

## IntVector.cpp

```
1  #include "IntVector.h"
2  #include <stdexcept>
3
4  IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements)
5  {
6      fNumberOfElements = aNumberOfElements;
7      fElements = new int[fNumberOfElements];
8
9      for (size_t i = 0; i < fNumberOfElements; i++)
10     {
11         fElements[i] = aArrayOfIntegers[i];
12     }
13 }
14
15 IntVector::~IntVector()
16 {
17     delete[] fElements;
18 }
19
20 size_t IntVector::size() const
21 {
22     return fNumberOfElements;
23 }
24
25 void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex)
26 {
27     if (aSourceIndex < fNumberOfElements && aTargetIndex < fNumberOfElements)
28     {
29         int temp = fElements[aSourceIndex];
30         fElements[aSourceIndex] = fElements[aTargetIndex];
31         fElements[aTargetIndex] = temp;
32         return;
33     }
34
35     throw std::out_of_range("Illegal vector indices");
36 }
37
38 const int IntVector::get(size_t aIndex) const
39 {
40     if (aIndex < fNumberOfElements)
41     {
42         return (*this)[aIndex];
43     }
44
45     throw std::out_of_range("Illegal vector index");
46 }
47
48 const int IntVector::operator[](size_t aIndex) const
49 {
50     if (aIndex < fNumberOfElements)
51     {
52         return fElements[aIndex];
53     }
54
55     throw std::out_of_range("Illegal vector index");
56 }
```

## SortableIntVector.cpp

```
1  #include "SortableIntVector.h"
2
3  SortableIntVector::SortableIntVector(const int aArrayOfIntegers[], size_t
aNumberOfElements) : IntVector::IntVector(aArrayOfIntegers, aNumberOfElements) { }
4
5  void SortableIntVector::sort(Comparable aOrderFunction) {
6      for (size_t i = size() - 1; i > 0; i--) {
7          for (size_t j = 0; j < i; j++) {
8              if (aOrderFunction(get(j), get(j + 1))) {
9                  swap(j, j + 1);
10             }
11         }
12     }
13 }
```

## ShakerSortableIntVector.cpp

```
1  #include "ShakerSortableIntVector.h"
2  #include <iostream>
3
4  using namespace std;
5
6  ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers[], size_t
aNumberOfElements) : SortableIntVector::SortableIntVector(aArrayOfIntegers,
aNumberOfElements) { }
7
8  void ShakerSortableIntVector::sort(Comparable aOrderFunction) {
9      size_t start = 0, end = size() - 1;
10
11      while (start < end) {
12          for (size_t i = start; i < end; i++) {
13              if (aOrderFunction(get(i), get(i + 1))) {
14                  swap(i, i + 1);
15              }
16          }
17          end--;
18
19          for (size_t i = end; i > start; i--) {
20              if (aOrderFunction(get(i - 1), get(i))) {
21                  swap(i - 1, i);
22              }
23          }
24          start++;
25      }
26  }
```

## Main\_PS2.cpp

```
1
2 // Problem Set 2, 2022
3
4 #include <iostream>
5 #include <stdexcept>
6
7 using namespace std;
8
9 #define P1
10 #define P2
11 #define P3
12
13 #ifdef P1
14
15 #include "IntVector.h"
16
17 void runP1()
18 {
19     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
20     size_t lArrayLength = sizeof(lArray) / sizeof(int);
21
22     IntVector lVector( lArray, lArrayLength );
23
24     cout << "Test range check:" << endl;
25
26     try
27     {
28         int lValue = lVector[lArrayLength];
29
30         cerr << "Error, you should not see " << lValue << " here!" << endl;
31     }
32     catch (out_of_range e)
33     {
34         cerr << "Properly caught error: " << e.what() << endl;
35     }
36     catch (...)
37     {
38         cerr << "This message must not be printed!" << endl;
39     }
40
41     cout << "Test swap:" << endl;
42
43     try
44     {
45         cout << "lVector[3] = " << lVector[3] << endl;
46         cout << "lVector[6] = " << lVector[6] << endl;
47
48         lVector.swap( 3, 6 );
49
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;
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     // Use a lambda expression here that orders integers in increasing order.
92     // The lambda expression does not capture any variables or throws any exceptions.
93     // It has to return a bool value.
94
95     auto sortFunction = [](int a, int b) -> bool {
96         return a > b;
97     };
98
99     lVector.sort( sortFunction );
100
101     cout << "After sorting:" << endl;
102
103     for ( size_t i = 0; i < lVector.size(); i++ )
104     {
105         cout << lVector[i] << ' ';
106     }
107
108     cout << endl;
109 }
110
111 #endif
112
113 #ifdef P3
114
115 #include "ShakerSortableIntVector.h"
116
117 void runP3()

```

```

118 {
119     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
120     size_t lArrayLength = sizeof(lArray) / sizeof(int);
121
122     ShakerSortableIntVector lVector( lArray, lArrayLength );
123
124     cout << "Cocktail Shaker Sort:" << endl;
125
126     cout << "Before sorting:" << endl;
127
128     for ( size_t i = 0; i < lVector.size(); i++ )
129     {
130         cout << lVector[i] << ' ';
131     }
132
133     cout << endl;
134
135     // sort in decreasing order
136     lVector.sort();
137
138     cout << "After sorting:" << endl;
139
140     for ( size_t i = 0; i < lVector.size(); i++ )
141     {
142         cout << lVector[i] << ' ';
143     }
144
145     cout << endl;
146 }
147
148 #endif
149
150 int main()
151 {
152     #ifdef P1
153
154         runP1();
155
156     #endif
157
158     #ifdef P2
159
160         runP2();
161
162     #endif
163
164     #ifdef P3
165
166         runP3();
167
168     #endif
169
170     return 0;
171 }
172

```