

**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: \_\_\_\_\_

```
1  #include "IntVector.h"
2  #include <stdexcept>
3
4  using namespace std;
5
6  IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements)
7  {
8      fNumberOfElements = aNumberOfElements;    // passing in the number of
          elements
9      fElements = new int[fNumberOfElements];    //creating an integer array
          fElements
10
11      for (size_t i = 0; i < fNumberOfElements; i++)
12      {
13          fElements[i] = aArrayOfIntegers[i];    //looping through and
          populating the array with values
14      }
15  }
16
17  IntVector::~IntVector()
18  {
19      delete[] fElements;    //destructor deletes the elements and array
20  }
21
22  size_t IntVector::size() const
23  {
24      return fNumberOfElements;    //gives back the size of the vector
25  }
26
27  const int IntVector::get(size_t aIndex) const
28  {
29      if (aIndex >= fNumberOfElements)
30      {
31          throw out_of_range("Illegal vector indices.");
32      }
33      return (*this)[aIndex];
34  }
35
36  void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex) //the member
          function swap() takes two indices and, if they are within range, swaps the
37
          //
          corresponding array elements in an IntVector object. We
          need swap() for sorting.
38  {
39      int temp = get(aSourceIndex);
40      fElements[aSourceIndex] = get(aTargetIndex);
41      fElements[aTargetIndex] = temp;
42
43
```

```
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
```

```
1
2 #include "SortableIntVector.h"
3
4 using namespace std;
5
6 SortableIntVector::SortableIntVector(const int aArrayOfIntegers[], size_t      ↗
    aNumberOfElements) : IntVector::IntVector(aArrayOfIntegers,      ↗
    aNumberOfElements)
7 {}
8
9 void SortableIntVector::sort(Comparable aOrderFunction)
10 {
11     for (size_t i = 0; i < size() - 1; i++)          //loop through size of the ↗
        array -1
12     {
13         for (size_t j = 0; j < size() - i - 1; j++) //inner loop
14         {
15             if (aOrderFunction(get(j), get(j + 1))) // if the comparable ↗
                returns true
16             {
17                 swap(j, j + 1);          //swap the values
18             }
19         }
20     }
21 }
22
23
24
25
26
27
```

```
1
2 #include "SortableIntVector.h"
3 #include "ShakerSortableIntVector.h"
4
5 using namespace std;
6
7
8
9 ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers[], ↗
    size_t aNumberOfElements) : SortableIntVector::SortableIntVector ↗
    (aArrayOfIntegers, aNumberOfElements)
10 {}
11
12
13 void ShakerSortableIntVector::sort(Comparable aOrderFunction)
14 {
15     int start = 0;
16     int end = size() - 1;
17     while (start < end)
18     {
19         for (int i = start; i < end; ++i)
20         {
21             if (aOrderFunction(get(i), get(i + 1)))    // if the comparable ↗
22                 returns true
23             {
24                 swap(i, i + 1);    //swap the values
25             }
26         }
27         --end;
28
29         for (int i = end - 1; i >= start; --i)
30         {
31             if (aOrderFunction(get(i), get(i + 1)))    // if the comparable ↗
32                 returns true
33             {
34                 swap(i, i + 1);    //swap the values
35             }
36         }
37         ++start;
38     }
39 }
```

```
1
2 // Problem Set 2, 2022
3
4 #include <iostream>
5 #include <stdexcept>
6 #include <array>
7 using namespace std;
8
9 #define P1
10 #define P2
11 #define P3
12
13 #ifndef 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; //this is a
        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     runP2();
152
153 #endif
154
155 #ifdef P3
156     runP3();
157
158 #endif
159
160 #endif
161
162     return 0;
163 }
164
```