

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

## Problem 1 – IntVector.cpp

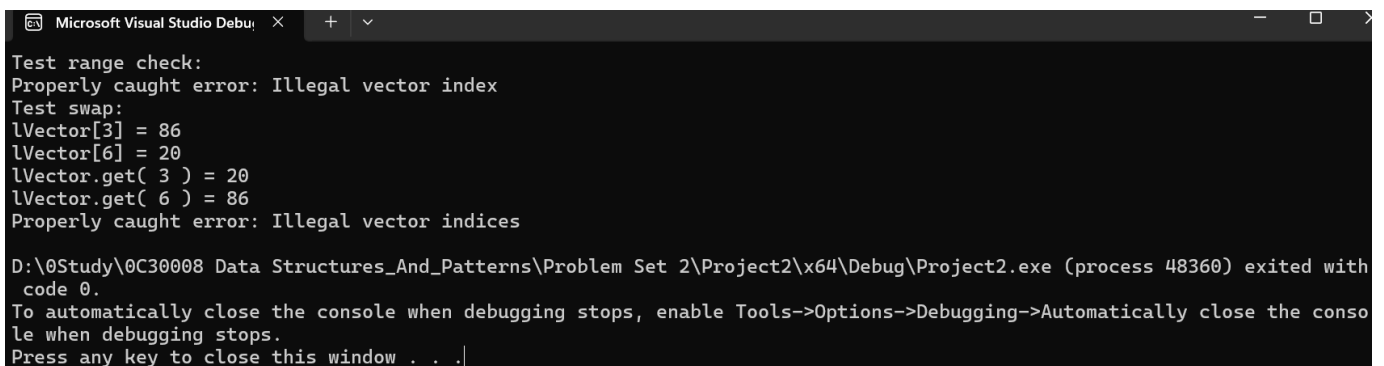
...atterns\Problem Set 2\Project2\Project2\IntVector.cpp

1

```
1  #include "IntVector.h"
2  #include <stdexcept> // Handle exceptions
3
4  using namespace std;
5
6  // Constructor: Initialize with an array of integers and its size
7  IntVector::IntVector(const int aArrayOfIntegers[], size_t aNumberOfElements)
8  {
9      fNumberOfElements = aNumberOfElements; // Set size
10     fElements = new int[fNumberOfElements]; // Allocate memory
11
12     // Copy elements to the internal array
13     for (size_t i = 0; i < fNumberOfElements; i++)
14     {
15         fElements[i] = aArrayOfIntegers[i];
16     }
17 }
18
19 // Destructor: Release allocated memory
20 IntVector::~IntVector()
21 {
22     delete[] fElements; // Free memory
23 }
24
25 // Return the size of the vector
26 size_t IntVector::size() const
27 {
28     return fNumberOfElements; // Return size
29 }
30
31 // Get the element at a specific index
32 const int IntVector::get(size_t aIndex) const
33 {
34     return (*this)[aIndex]; // Return element at index
35 }
36
37 // Swap two elements by their indices
38 void IntVector::swap(size_t aSourceIndex, size_t aTargetIndex)
39 {
40     // Ensure indices are within bounds
41     if (aSourceIndex < fNumberOfElements && aTargetIndex <
42         fNumberOfElements)
43     {
44         int lTemp = fElements[aSourceIndex]; // Temp store
45         fElements[aSourceIndex] = fElements[aTargetIndex]; // Swap
46         fElements[aTargetIndex] = lTemp; // Complete swap
47     }
48     else
49     {
50         // Handle out-of-bounds indices
51     }
52 }
```

```
48     {
49         throw out_of_range("Illegal vector indices"); // Handle out-of-
           bounds access
50     }
51 }
52
53 // Overload [] to access elements at a given index
54 const int IntVector::operator[](size_t aIndex) const
55 {
56     if (aIndex < fNumberOfElements)
57     {
58         return fElements[aIndex]; // Return element
59     }
60
61     throw out_of_range("Illegal vector index"); // Handle invalid index
62 }
63
```

## Output:



Microsoft Visual Studio Debug Console output:

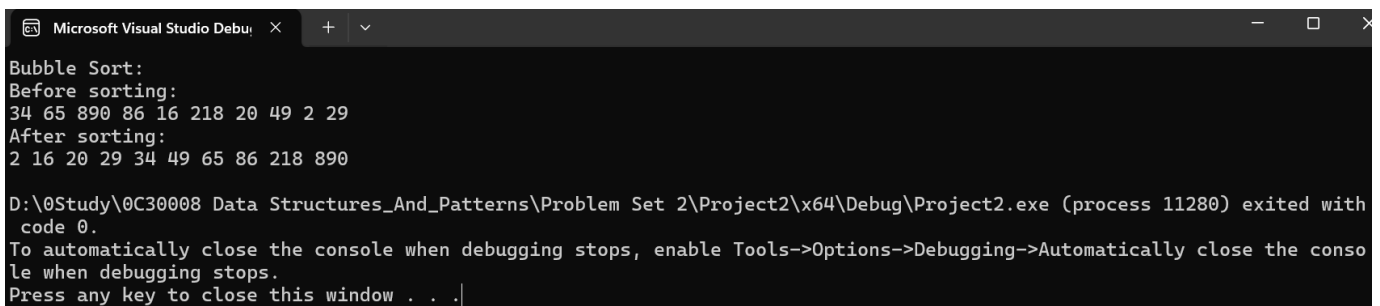
```
Test range check:
Properly caught error: Illegal vector index
Test swap:
lVector[3] = 86
lVector[6] = 20
lVector.get( 3 ) = 20
lVector.get( 6 ) = 86
Properly caught error: Illegal vector indices

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 2\Project2\x64\Debug\Project2.exe (process 48360) exited with
code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .|
```

## Problem 2 – SortableIntVector.cpp

```
...Problem Set 2\Project2\Project2\SortableIntVector.cpp 1
1 #include "SortableIntVector.h"
2
3 // Constructor: Initialize using base class IntVector
4 SortableIntVector::SortableIntVector(const int aArrayOfIntegers[], size_t ↗
    aNumberOfElements) :
5     IntVector(aArrayOfIntegers, aNumberOfElements)
6 {}
7
8 // Sort function using a comparison function to define the order
9 void SortableIntVector::sort(Comparable aOrderFunction)
10 {
11     // Outer loop through all elements
12     for (size_t i = 0; i < size(); i++)
13     {
14         // Inner loop to compare and swap elements
15         for (size_t j = size() - 1; j > i; j--)
16         {
17             if (aOrderFunction(get(j), get(j - 1))) // Compare elements
18             {
19                 swap(j, j - 1); // Swap elements if needed
20             }
21         }
22     }
23 }
24
```

## Output:



The screenshot shows a Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

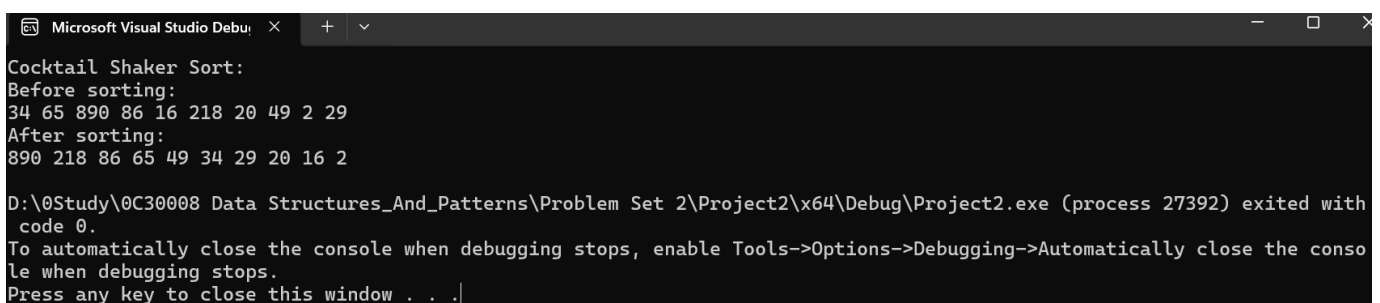
```
Bubble Sort:
Before sorting:
34 65 890 86 16 218 20 49 2 29
After sorting:
2 16 20 29 34 49 65 86 218 890

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 2\Project2\x64\Debug\Project2.exe (process 11280) exited with
code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .|
```

## Problem 3 – ShakerSortableIntVector.cpp

```
...m Set 2\Project2\Project2\ShakerSortableIntVector.cpp 1
1 #include "ShakerSortableIntVector.h"
2
3 // Constructor, initializes using base class constructor
4 ShakerSortableIntVector::ShakerSortableIntVector(const int aArrayOfIntegers >
    [], size_t aNumberOfElements) :
5     SortableIntVector(aArrayOfIntegers, aNumberOfElements)
6 {}
7
8 // Shaker sort function, sorts elements using a comparison function
9 void ShakerSortableIntVector::sort(Comparable aOrderFunction)
10 {
11     size_t lBeginIndex = 0; // Start of the unsorted section
12     size_t lEndIndex = size() - 1; // End of the unsorted section
13
14     // Continue sorting while there are unsorted elements
15     while (lBeginIndex < lEndIndex)
16     {
17         // Forward pass to push the largest element to the end
18         for (size_t i = lBeginIndex; i < lEndIndex; i++)
19         {
20             if (aOrderFunction(get(i), get(i + 1)))
21             {
22                 swap(i, i + 1); // Swap adjacent elements
23             }
24         }
25
26         lEndIndex--; // Reduce the unsorted section from the end
27
28         // Backward pass to push the smallest element to the beginning
29         for (size_t i = lEndIndex; i > lBeginIndex; i--)
30         {
31             if (aOrderFunction(get(i - 1), get(i)))
32             {
33                 swap(i - 1, i); // Swap adjacent elements
34             }
35         }
36
37         lBeginIndex++; // Reduce the unsorted section from the beginning
38     }
39 }
40
```

## Output:



```
Microsoft Visual Studio Debug Console
Cocktail Shaker Sort:
Before sorting:
34 65 890 86 16 218 20 49 2 29
After sorting:
890 218 86 65 49 34 29 20 16 2

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 2\Project2\x64\Debug\Project2.exe (process 27392) exited with
code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window .. .|
```

## Main\_PS2 – line 93 added

...Patterns\Problem Set 2\Project2\Project2\Main\_PS2.cpp

1

```
1 #include <iostream>
2 #include <stdexcept>
3
4 using namespace std;
5
6 //define P1
7 //define P2
8 #define P3
9
10 #ifdef P1
11
12 #include "IntVector.h"
13
14 void runP1()
15 {
16     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
17     size_t lArrayLength = sizeof(lArray) / sizeof(int);
18
19     IntVector lVector( lArray, lArrayLength );
20
21     cout << "Test range check:" << endl;
22
23     try
24     {
25         int lValue = lVector[lArrayLength];
26
27         cerr << "Error, you should not see " << lValue << " here!" <<
28             endl;
29     }
30     catch (out_of_range e)
31     {
32         cerr << "Properly caught error: " << e.what() << endl;
33     }
34     catch (...)
35     {
36         cerr << "This message must not be printed!" << endl;
37     }
38
39     cout << "Test swap:" << endl;
40
41     try
42     {
43         cout << "lVector[3] = " << lVector[3] << endl;
44         cout << "lVector[6] = " << lVector[6] << endl;
45
46         lVector.swap( 3, 6 );
47
48         cout << "lVector.get( 3 ) = " << lVector.get( 3 ) << endl;
49         cout << "lVector.get( 6 ) = " << lVector.get( 6 ) << endl;
```

```
49
50     lVector.swap( 5, 20 );
51
52     cerr << "Error, you should not see this message!" << endl;
53 }
54 catch (out_of_range e)
55 {
56     cerr << "Properly caught error: " << e.what() << endl;
57 }
58 catch (...)
59 {
60     cerr << "Error, this message must not be printed!" << endl;
61 }
62 }
63
64 #endif
65
66 #ifdef P2
67
68 #include "SortableIntVector.h"
69
70 void runP2()
71 {
72     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
73     size_t lArrayLength = sizeof(lArray) / sizeof(int);
74
75     SortableIntVector lVector( lArray, lArrayLength );
76
77     cout << "Bubble Sort:" << endl;
78
79     cout << "Before sorting:" << endl;
80
81     for ( size_t i = 0; i < lVector.size(); i++ )
82     {
83         cout << lVector[i] << ' ';
84     }
85
86     cout << endl;
87
88     // Use a lambda expression here that orders integers in increasing order.
89     // The lambda expression does not capture any variables or throws any exceptions.
90     // It has to return a bool value.
91     // lVector.sort( /* lambda expression */ );
92
93     lVector.sort( [](int aLeft, int aRight) { return aLeft <= aRight; } ); // new line
94
```

```
95     cout << "After sorting:" << endl;
96
97     for ( size_t i = 0; i < lVector.size(); i++ )
98     {
99         cout << lVector[i] << ' ';
100     }
101
102     cout << endl;
103 }
104
105 #endif
106
107 #ifdef P3
108
109 #include "ShakerSortableIntVector.h"
110
111 void runP3()
112 {
113     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
114     size_t lArrayLength = sizeof(lArray) / sizeof(int);
115
116     ShakerSortableIntVector lVector( lArray, lArrayLength );
117
118     cout << "Cocktail Shaker Sort:" << endl;
119
120     cout << "Before sorting:" << endl;
121
122     for ( size_t i = 0; i < lVector.size(); i++ )
123     {
124         cout << lVector[i] << ' ';
125     }
126
127     cout << endl;
128
129     // sort in decreasing order
130     lVector.sort();
131
132     cout << "After sorting:" << endl;
133
134     for ( size_t i = 0; i < lVector.size(); i++ )
135     {
136         cout << lVector[i] << ' ';
137     }
138
139     cout << endl;
140 }
141
142 #endif
143
```



```
144 int main()
145 {
146     #ifdef P1
147
148         runP1();
149
150     #endif
151
152     #ifdef P2
153
154         runP2();
155
156     #endif
157
158     #ifdef P3
159
160         runP3();
161
162     #endif
163
164     return 0;
165 }
166
```