

```
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!" <<      ↗
31             endl;
32     }
33     catch (out_of_range e)
34     {
35         cerr << "Properly caught error: " << e.what() << endl;
36     }
37     catch (...)
38     {
39         cerr << "This message must not be printed!" << endl;
40     }
41
42     cout << "Test swap:" << endl;
43
44     try
45     {
46         cout << "lVector[3] = " << lVector[3] << endl;
47         cout << "lVector[6] = " << lVector[6] << endl;
48
49         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 ➤
93     // exceptions.
94     // It has to return a bool value.
95     lVector.sort([](const int aLHS, const int aRHS) -> bool
96     {
```

```
96         return aLHS <= aRHS;
97     });
98
99     cout << "After sorting:" << endl;
100
101     for ( size_t i = 0; i < lVector.size(); i++ )
102     {
103         cout << lVector[i] << ' ';
104     }
105
106     cout << endl;
107 }
108
109 #endif
110
111 #ifdef P3
112
113 #include "ShakerSortableIntVector.h"
114
115 void runP3()
116 {
117     int lArray[] = { 34, 65, 890, 86, 16, 218, 20, 49, 2, 29 };
118     size_t lArrayLength = sizeof(lArray) / sizeof(int);
119
120     ShakerSortableIntVector lVector( lArray, lArrayLength );
121
122     cout << "Cocktail Shaker Sort:" << endl;
123
124     cout << "Before sorting:" << endl;
125
126     for ( size_t i = 0; i < lVector.size(); i++ )
127     {
128         cout << lVector[i] << ' ';
129     }
130
131     cout << endl;
132
133     // sort in decreasing order
134     lVector.sort();
135
136     cout << "After sorting:" << endl;
137
138     for ( size_t i = 0; i < lVector.size(); i++ )
139     {
140         cout << lVector[i] << ' ';
141     }
142
143     cout << endl;
144 }
```

```
145
146 #endif
147
148 int main()
149 {
150     #ifdef P1
151
152         runP1();
153
154     #endif
155
156     #ifdef P2
157
158         runP2();
159
160     #endif
161
162     #ifdef P3
163
164         runP3();
165
166     #endif
167
168     return 0;
169 }
170
```