```
 1: // SCSJ1023, Programming Technique II
 2: // Semester 1, 2017/2018
 3: //
 4: // Final Exam, Paper 2
 5: // Program 1
 6: // Structured programming question
 7: // *** SOLUTION ***
 8:
 9: #include <iostream>
10: #include <exception>
11:
12: using namespace std;
13: const int MAX = 3;   // The maximum number of elements the array can hold
14:
15: class Array
16: {
17:     private:
18:         int data[MAX];   // array elements
19:         int count;       // the number of element currently held by the array
20:
21:     public:
22:
23:         // Task 1: Define three exception classes named 'Full', 'Empty' and 'NegativeIndex
24:
25:         class Full{};
26:         class Empty{};
27:         class NegativeIndex{};
28:
29:         Array(){ count = 0;}
30:         int getCount() const {return count;}
31:
32:         // Method add: To add an element to the array
33:         void add(int element)
34:         {
35:            // Task 2: Throw a  'Full' exception if the array already holds the maximum num
36:            if(count >= MAX)
37:                 throw Full();
38:
39:            data[count] = element;
40:            count++;
41:         }
42:
43:         // To remove an element from the array.
44:         void remove()
45:         {
46:            // Task 3: Throw an 'Empty' exception if the array is empty, i.e., no item held
47:            if(count <= 0)
48:              throw Empty();
49:
50:            count--;
51:         }
52:
53:
54:         // Method displayElement: To display an element based on its index entered from th
55:         void displayElement()
56:         {
57:             int index;
58:
59:             cout << "Enter the index of the element you want to display => ";
60:             cin >> index;
```

```
61:
62:             // Task 4: Throw a 'NegativeIndex' exception if the user entered a negative va
63:
64:             if (index < 0)
65:                 throw NegativeIndex();
66:
67:             // Task 5: Throw the index value entered by the user if the value is larger th
68:
69:             if (index >= count)
70:                 throw index;
71:
72:
73:             cout << "Index: " << index << ", Element: " << data[index] << endl;
74:         }
75: };
76:
77:
78: int main()
79: {
80:     Array a;
81:
82:     a.add(11);
83:     cout << "Number 11 has been added. Current number of element = "  << a.getCount() << e
84:
85:     a.add(22);
86:     cout << "Number 22 has been added. Current number of element = "  << a.getCount() << e
87:
88:     cout << endl;
89:     try{
90:         a.displayElement();
91:     }
92:
93:     // Task 6: Handle the case where the user has entered a negative index. See the ex
94:     catch (Array::NegativeIndex) {
95:         cout << "Error! You have entered a negative index." << endl;
96:     }
97:
98:     // Task 7: Handle the case where the user has entered the index that is larger tha
99:     catch (int e){
100:         cout << "Error! You have entered index value of " << e << endl;
101:         cout << " while the current number of elements is  " << a.getCount() << endl;
102:     }
103:
104:     catch (...){}
105:
106:     cout << endl;
107:     try{
108:         a.add(33);
109:         cout << "Number 33 has been added. Current number of element = "  << a.getCount()
110:
111:         a.add(44);
112:         cout << "Number 44 has been added. Current number of element = "  << a.getCount()
113:     }
114:
115:     // Task 8: Handle the case where an element wants to be added but the array is alr
116:
117:     catch (Array::Full) {
118:         cout << "The array is full." << endl;
119:     }
120:
```

```
121:     catch (...){}
122:
123:     cout << endl;
124:     try{
125:         a.remove();
126:         cout << "An element has been removed. Current number of element = "  << a.getCount
127:
128:         a.remove();
129:         cout << "An element has been removed. Current number of element = "  << a.getCount
130:
131:         a.remove();
132:         cout << "An element has been removed. Current number of element = "  << a.getCount
133:
134:         a.remove();
135:         cout << "An element has been removed. Current number of element = "  << a.getCount
136:     }
137:
138:     // Task 9: Handle the case where an element wants to be removed but the array is e
139:     catch (Array::Empty) {
140:         cout << "The array is empty" << endl;
141:     }
142:
143:     catch (...){}
144:
145:     return 0;
146:
147: }
```