```
1 // Justin Dang Student ID: 1148267
2 /*
 3 creates a LinkedQueue (int)array
4 */
 5 #include <iostream>
 6 using namespace std;
7 #define MaxSize 10
10 class arrayQueue {
11 private:
        int front, // holds node number of the front node(0)
12
            back, // holds node number of the back node(x) x = any number between →
13
              0-10
14
           nodeCount;
                           // holds the amount of nodes created
       int queue[MaxSize]; // array where the ints will be stored
15
16 public:
17
        arrayQueue() { front = back = nodeCount = 0; } // queue contstructor
18
       bool isEmpty() { return nodeCount == 0; } // returns true if there are
          no nodes
       bool enqueue(int data) {
19
           if (nodeCount++ == MaxSize) { // catches error if the node count is 1
20
              less than
                nodeCount--;
21
                                          // MaxSize of our array
                cout << "Maximum queue size hit.\n\n";</pre>
22
                cout << "Front: " << front << endl</pre>
23
                    << "Back: " << back << endl</pre>
24
25
                    << "nodeCount: " << nodeCount << "\n\n";</pre>
26
                return false;
27
            }
28
29
           queue[back++] = data; // queues the integer at the back of the array
30
31
            if (back >= MaxSize) // used to wrap back to the front of the array
32
                back = 0:
           return true;
33
34
       }
       int dequeue() {
35
            if (isEmpty()) { // catches error if the amount of nodes we have are 0
36
37
                cout << "Queue is empty.\n\n";</pre>
38
                return -999;
39
           }
40
41
           nodeCount--;
                                            // removes 1 from node count to reflect
              action
42
           int intReturn = queue[front++]; // stores the front int into a temp
43
44
           if (front > MaxSize)
                                    // rotates the queue to the front of
              array
45
                front = 0;
46
47
           return intReturn;
```

```
...Justin Dang\Desktop\Data Structures\Array Based Queue.cpp
```

```
48
       void print() {
49
50
            cout << "From front to back, the queue values are: ";</pre>
51
            for (int i = front; i < back; i++)</pre>
52
                cout << queue[i] << ' ';</pre>
53
            cout << "\n\n";</pre>
54
       }
55 };
56 int main()
57 {
       cout << "Creating a linkedQueue array. \n\n\n";</pre>
58
59
       arrayQueue *array1;
60
       array1 = new arrayQueue();
61
       cout << "Enqueue 15.\n\n";</pre>
62
       array1->enqueue(15);
63
       array1->print();
64
65
       cout << "Enqueue 3.\n\n";</pre>
66
       array1->enqueue(3);
67
       array1->print();
68
       cout << "Enqueue 5.\n\n";</pre>
69
70
       array1->enqueue(5);
71
       array1->print();
72
73
       cout << "Enqueue 10.\n\n";</pre>
74
       array1->enqueue(10);
75
       array1->print();
76
       cout << "----\nEmptying Queue. . .\n\n";</pre>
77
78
       // removes every node in the array until an error is thrown
79
       do{
            cout << "Removed " << array1->dequeue() << " from the array." << "\n\n";</pre>
80
81
            array1->print();
82
       } while (!array1->isEmpty());
83
        array1->dequeue();
84
85
       // fills every node in the array until error is thrown
       cout << "-----Filling queue. . .\n\n";</pre>
86
87
       while (array1->enqueue(5))
            cout << '.';
88
89
       array1->print();
90 }
91 /*//-----case 1:
92 Creating a linkedQueue array.
93
94
95 Enqueue 15.
96
97 From front to back, the queue values are: 15
98
99 Enqueue 3.
```

```
100
101 From front to back, the queue values are: 15 3
102
103 Enqueue 5.
104
105 From front to back, the queue values are: 15 3 5
106
107 Enqueue 10.
108
109 From front to back, the queue values are: 15 3 5 10
110
111 -----
112 Emptying Queue. . .
113
114 Removed 15 from the array.
115
116 From front to back, the queue values are: 3 5 10
117
118 Removed 3 from the array.
119
120 From front to back, the queue values are: 5 10
121
122 Removed 5 from the array.
123
124 From front to back, the queue values are: 10
125
126 Removed 10 from the array.
127
128 From front to back, the queue values are:
129
130 Queue is empty.
131
132 -----Filling queue. . .
133
134 ......Maximum queue size hit.
135
136 Front: 4
137 Back: 4
138 nodeCount: 10
140 From front to back, the queue values are:
141
142
143 *///-----
144
145
```