```
1 // Justin Dang Student ID: 1148267
2 /*
 3 Creates an link based Queue using enqueue and dequeue methods
 5 Queue is size "unlimited" only limited by computer hardware
7 Error is thrown when attempting to dequeue nothing from the list
9 only takes in data type int for each node list
10 */
11 #include <iostream>
12 using namespace std;
13
14 class Node {
15 public:
16
       int data;
                                   // data in each node
       class Node* next;
                          // address of next node(or null/0 to define as end of
17
         Queue)
       Node(int info, Node* ptr = 0) { // Structure for each node
18
19
           data = info;
20
           next = ptr;
21
       }
22 };
23
24 class LinkedQueue {
25 public:
26
       LinkedQueue() { front = back = 0; } // constructor that is used to ensure
         our first node and last are set properly
27
       bool isEmpty() { return front == 0; } // checks if our first node has a
         address(implying there is another node)
       void enqueue(int info) {
28
29
           Node* temp = new Node(info); // new node stored in temp(note no address
             given to show it is last node)
30
           nodeCount++;
                                        // increase node count
31
           if (back == 0)
               front = back = temp; // if our last node is null/0 then we create →
32
                   a node that is the front and back
33
           else
34
           {
35
               back->next = temp;
                                         // otherwise we set the address of our last >
                 node to our new node
36
                                         // set our last node = new node
               back = temp;
37
           }
38
       }
       int dequeue() {
39
40
           if (isEmpty()) {
                                        // if our queue is empty then throw error
41
               cout << "The Queue is empty.\n\n";</pre>
42
               nodeCount++;
                                        // offset the nodeCount decrement
43
               return -999;
44
           }
45
           nodeCount--;
           Node* temp;
                                        // temp node where our front node goes
46
```

```
...\Justin Dang\Desktop\Data Structures\Link Based Queue.cpp
                                                                                           2
47
             int frontInt = front->data; // we take the data from the front node
48
             temp = front;
             front = front->next;
                                           // our element after the first element is
49
               now the front element
50
             return frontInt;
                                           // we return data for from our front node
 51
         }
52
         void print() {
             cout << "There are " << nodeCount << " items in the Queue." << "\nThe</pre>
53
                                                                                           P
               queue from first to last is: ";
54
             for (Node* temp = front; temp != 0; temp = temp->next)
                 cout << temp->data << " ";// All queue data is printed</pre>
55
56
             cout << "\n\n";</pre>
57
         }
58 private:
59
         Node* front, * back;
60
         int nodeCount;
61 };
62 int main()
63 {
64
         cout << "Creating Link Based Queue. . .\n\n";</pre>
65
         LinkedQueue* ptr = new LinkedQueue(); // New queue is started
66
         cout << "Enqueue(15)\n\n";</pre>
                                                // begins adding ints to queue
67
68
         ptr->enqueue(15);
69
         ptr->print();
70
         cout << "Enqueue(5)\n\n";</pre>
71
72
         ptr->enqueue(5);
73
         ptr->print();
74
75
         cout << "Enqueue(20)\n\n";</pre>
76
         ptr->enqueue(20);
77
         ptr->print();
78
         cout << "Enqueue(10)\n\n";</pre>
79
80
         ptr->enqueue(10);
81
         ptr->print();
82
         cout << "Enqueue(25)\n\n";</pre>
83
84
         ptr->enqueue(25);
85
         ptr->print();
86
87
         cout << "Enqueue(35)\n\n";</pre>
88
         ptr->enqueue(35);
89
         ptr->print();
90
         cout << "----\n\n";
91
92
         cout << "Removing all nodes from queue. . .\n\n";</pre>
93
         while (!ptr->isEmpty()) {
                                                                                   //
                                                                                           P
           while the queue is not empty
94
             cout << "Removing " << ptr->dequeue() << " from the queue. \n\n"; //</pre>
```

remove an item

```
...\Justin Dang\Desktop\Data Structures\Link Based Queue.cpp
                                                                                 3
     ptr->print();
 96
        }
                                                                          //
 97
        ptr->dequeue();
                                                                                 P
          attempt to remove null to show error
98 }
99 /*// ----- case 1:
100 Creating Link Based Queue. . .
101
102 Enqueue(15)
103
104 There are 1 items in the Queue.
105 The queue from first to last is: 15
106
107 Enqueue(5)
108
109 There are 2 items in the Queue.
110 The queue from first to last is: 15 5
111
112 Enqueue(20)
113
114 There are 3 items in the Queue.
115 The queue from first to last is: 15 5 20
116
117 Enqueue(10)
118
119 There are 4 items in the Queue.
120 The queue from first to last is: 15 5 20 10
121
122 Enqueue(25)
123
124 There are 5 items in the Queue.
125 The queue from first to last is: 15 5 20 10 25
126
127 Enqueue(35)
128
129 There are 6 items in the Queue.
130 The queue from first to last is: 15 5 20 10 25 35
131
132 -----
134 Removing all nodes from queue. . .
135
136 Removing 15 from the queue.
137
138 There are 5 items in the Queue.
139 The queue from first to last is: 5 20 10 25 35
140
141 Removing 5 from the queue.
142
143 There are 4 items in the Queue.
144 The queue from first to last is: 20 10 25 35
145
```

```
146 Removing 20 from the queue.
147
148 There are 3 items in the Queue.
149 The queue from first to last is: 10 25 35
150
151 Removing 10 from the queue.
152
153 There are 2 items in the Queue.
154 The queue from first to last is: 25 35
155
156 Removing 25 from the queue.
157
158 There are 1 items in the Queue.
159 The queue from first to last is: 35
160
161 Removing 35 from the queue.
162
163 There are 0 items in the Queue.
164 The queue from first to last is:
165
166 The Queue is empty.
```

167 */// -----