```
 1: /*Copyright [2016] <Albara Mehene> */
 2:
 3: #include "RingBuffer.hpp"
 4:
 5: // creates a empty ringbuffer with a max capacity
 6: RingBuffer::RingBuffer(int capacity) {
 7: if(capacity < 1) {
 8:     throw std::invalid_argument("capacity must be greater than zero");
 9: }
10:     cap = capacity;
11:     count = 0;
12:     array = new int16_t[capacity];
13:     first = array;
14:     last = array;
15: }
16:
17: // returns the number of items currently in the buffer
18: int RingBuffer::size() {
19:     return count;
20: }
21:
22: // checks to see if the buffer is empty
23: bool RingBuffer::isEmpty() {
24:     if (count == 0) {
25:         return 1;
26:     } else {
27:         return 0;
28:     }
29: }
30:
31: // checks to see if the buffer is full
32: bool RingBuffer::isFull() {
33:     if (count == cap) {
34:        return 1;
35:     } else {
36:        return 0;
37:     }
38: }
39:
40: // add item x to the end of buffer
41: void RingBuffer::enqueue(int16_t x) {
42: if(isFull() == 1) {
43:     throw std::runtime_error("can't enqueue to a full ring");
44: }
45:
46:     if (last == (array+(cap-1))){
47:          count++;
48:          (*last) = x;
49:          last = array;
50:     }else{
51:         count++;
52:         (*last) = x;
53:         last = (last + 1);
54:     }
55: }
56:
57: // deletes and returns the item from the front of the buffer
58: int16_t RingBuffer::dequeue() {
59:     if (isEmpty() == 1) {
60:        throw std::runtime_error("can't dequeue to a empty ring");
61:     }
```

```
62:
63:     int16_t store;
64:
65:     if(first == (array+(cap-1))){
66:       count--;
67:       store = (*first);
68:       first = array;
69:       return store;
70:     }else{
71:         count--;
72:         store = (*first);
73:         first = (first+1);
74:         return store;
75:     }
76: }
77:
78: // returns item from the front without deleting it
79: int16_t RingBuffer::peek() {
80:     int16_t temp = (*first);
81:     return temp;
82: }
83:
84: RingBuffer::~RingBuffer() {
85:     delete[] array;
86: }
87: void RingBuffer::print_out(){
88:
89:     for(int i = 0; i < cap;i++){
90:         std::cout << "Array:   " << array[i] << std::endl;
91:     }
92: }
```