

```
1: /*Copyright 2017 Sam Pickell*/
2: #include "RingBuffer.hpp"
3:
4: RingBuffer::RingBuffer() {
5:     my_size = 0;
6:     capacity = 1;
7: }
8:
9: RingBuffer::RingBuffer(int u_capacity) {
10:     my_size = 0;
11:
12:     if (u_capacity < 1) {
13:         throw std::invalid_argument(
14:             "RB constructor: capacity must be greater than zero.");
15:     } else {
16:         capacity = u_capacity;
17:     }
18: }
19:
20: RingBuffer::~RingBuffer() {}
21:
22: bool RingBuffer::isEmpty() {
23:     return (my_size == 0);
24: }
25:
26: bool RingBuffer::isFull() {
27:     return (my_size == capacity);
28: }
29:
30: void RingBuffer::enqueue(int16_t x) {
31:     if (this->isFull()) {
32:         throw std::runtime_error("enqueue: can't enqueue to a full ring.");
33:     } else {
34:         data.push_back(x);
35:         my_size++;
36:     }
37: }
38:
39: int16_t RingBuffer::dequeue() {
40:     int16_t temp;
41:     if (this->isEmpty()) {
42:         throw std::runtime_error("dequeue: nothing to dequeue, empty.");
43:     } else {
44:         temp = data.front();
45:         data.erase(data.begin());
46:         my_size--;
47:     }
48:     return temp;
49: }
50:
51: int16_t RingBuffer::peek() {
52:     if (this->isEmpty()) {
53:         throw std::runtime_error("peek: nothing to see, empty.");
54:     }
55:     return data.front();
56: }
```