```cpp
/*Copyright 2017 Sam Pickell*/
#include "RingBuffer.hpp"

RingBuffer::RingBuffer() {
  my_size = 0;
  capacity = 1;
}

RingBuffer::RingBuffer(int u_capacity) {
  my_size = 0;

  if (u_capacity < 1) {
       throw std::invalid_argument(
            "RB constructor: capacity must be greater than zero.");
    } else {
       capacity = u_capacity;
    }
}

RingBuffer::~RingBuffer() {}

bool RingBuffer::isEmpty() {
return (my_size == 0);
}

bool RingBuffer::isFull() {
return (my_size == capacity);
}

void RingBuffer::enqueue(int16_t x) {
  if (this->isFull()) {
      throw std::runtime_error("enqueue: can't enqueue to a full ring.");
    } else {
      data.push_back(x);
      my_size++;
    }
}

int16_t RingBuffer::dequeue() {
  int16_t temp;
  if (this->isEmpty()) {
      throw std::runtime_error("dequeue: nothing to dequeue, empty.");
    } else {
      temp = data.front();
      data.erase(data.begin());
      my_size--;
    }
  return temp;
}

int16_t RingBuffer::peek() {
  if (this->isEmpty()) {
      throw std::runtime_error("peek: nothing to see, empty.");
    }
  return data.front();
}
```