```
    1: /*
    2: Copyright 2019 Adam Baptista
    3:
    4:
    5: */
    6:
    7: #include "RingBuffer.hpp"
    8:
    9: RingBuffer::RingBuffer(int capacity) {
   10:     try {
   11:         if (capacity < 1)
   12:             throw invalid_argument
   13:                 ("Capacity cannot be less than or equal to 1.");
   14:     } catch(invalid_argument& e) {
   15:         cerr << "RB constuctor: capacity cannot be less than or equal to 1."
;
   16:         cerr << endl;
   17:         throw e;
   18:     }
   19:     //sets a certain amount of space for items, like vector = new int[capaci
ty];
   20:     vector.reserve(capacity);
   21:     for (int i = 0; i < capacity; i++)
   22:         vector.push_back(0);
   23:     size = 0;
   24:     first = 0;
   25:     last = 0;
   26:     this->capacity = capacity;
   27: }
   28:
   29: int RingBuffer::ringSize() {
   30:     return size;
   31: }
   32:
   33: bool RingBuffer::isEmpty() {
   34:     if (size > 0)
   35:         return false;
   36:     return true;
   37: }
   38:
   39: bool RingBuffer::isFull() {
   40:     if (capacity > size)
   41:         return false;
   42:     return true;
   43: }
   44:
   45: void RingBuffer::enqueue(int16_t x) {
   46:     try {
   47:         if (isFull())
   48:             throw runtime_error
   49:                 ("Enqueue: can’t enqueue an full ring");
   50:     } catch (runtime_error& e) {
   51:         cerr << "Enqueue: can’t enqueue an full ring";
   52:         cerr << endl;
   53:         throw e;
   54:     }
   55:     vector[last] = x;
   56:     if (last == capacity - 1) {
   57:         last = 0;
   58:     } else {
   59:         last++;
```

```
    }
    size++;
}

int16_t RingBuffer::dequeue() {
    try {
        if (isEmpty())
            throw runtime_error
                ("Dequeue: can't dequeue en empty buffer");
    } catch (runtime_error& e) {
        cerr << "Dequeue: can't dequeue en empty buffer";
        cerr << endl;
        throw e;
    }
    temp_first = peek();
    if (first == capacity - 1) {
        first = 0;
    } else {
        first++;
    }
    size--;
    return temp_first;
}

int16_t RingBuffer::peek() {
    try {
        if (isEmpty())
            throw runtime_error
                ("Peek: can't peek at an empty ring");
    } catch (runtime_error& e) {
        cerr << "Peek: can't peek at an empty ring";
        cerr << endl;
        throw e;
    }
    return vector[first];
}

/*
void RingBuffer::print() {
    for (int i = 0; i < capacity; i++)
        cout << vector[i] << endl;
    cout << endl;
}
*/

void RingBuffer::empty() {
    size = 0;
    first = 0;
    last = 0;
}
```