```
    1: /*
    2:   Copyright 2022 Matthew Lorette Anaya, matthew_loretteanaya@student.uml.
edu
    3: */
    4:
    5: #include "CircularBuffer.hpp"
    6:
    7: CircularBuffer::CircularBuffer(int capacity) {
    8:     if (capacity < 1) {
    9:         throw std::invalid_argument
   10:          ("Circular Buffer constructor: capacity must be greater than zer
o");
   11:     }
   12:
   13:     last = 0;
   14:     first = 0;
   15:     s = 0;
   16:     cap = capacity;
   17:     buff.resize(capacity);
   18:
   19:     return;
   20: }
   21:
   22: int CircularBuffer::size() {
   23:     return s;
   24: }
   25:
   26: bool CircularBuffer::isEmpty() {
   27:     if (s != 0) {
   28:         return false;
   29:     } else {
   30:       return true;
   31:     }
   32: }
   33:
   34: bool CircularBuffer::isFull() {
   35:     if (s == cap) {
   36:         return true;
   37:     } else {
   38:       return false;
   39:     }
   40: }
   41: void CircularBuffer::enqueue(int16_t x) {
   42:     if (isFull()) {
   43:         throw std::runtime_error("enqueue: can't enqueue to a full buffer
");
   44:     }
   45:     if (last >= cap) {
   46:         last = 0;
   47:     }
   48:     // Continue
   49:     buff.at(last) = x;
   50:     last++;
   51:     s++;
   52: }
   53:
   54: int16_t CircularBuffer::dequeue() {
   55:     if (isEmpty()) {
   56:         throw std::runtime_error("dequeue: can't dequeue an empty buffer"
);
   57:     }
   58:     int16_t retFirst = buff.at(first);
   59:     buff.at(first) = 0;
   60:     first++;
   61:     s--;
```

```
   62:
   63:     if (first >= cap) {
   64:         first = 0;
   65:     }
   66:
   67:     return retFirst;
   68: }
   69:
   70: int16_t CircularBuffer::peek() {
   71:     if (isEmpty()) {
   72:         throw std::runtime_error("peek: can't peek an empty buffer");
   73:     }
   74:     return buff.at(first);
   75: }
   76:
   77: void CircularBuffer::output() {
   78:     std::cout << "   First: " << first << "\n";
   79:     std::cout << "    Last: " << last << "\n";
   80:     std::cout << "Capacity: " << cap << "\n";
   81:     std::cout << "    Size: " << s << "\n";
   82:     std::cout << "Vector size: " << buff.size() << "\n";
   83:     std::cout << "Vector capacity: " << buff.capacity() << "\n";
   84:     std::cout << "Buffer (no blanks): \n";
   85:
   86:     int x = 0;
   87:     int y = first;
   88:
   89:     while (x < s) {
   90:         // Make the loop go back to 0 to continue printing.
   91:         if (y >= cap) {
   92:             y = 0;
   93:         }
   94:
   95:         std::cout << buff[y] << " ";
   96:         y++;
   97:         x++;
   98:     }
   99:
  100:     std::cout << "\nDump the entire buffer (including blanks): \n";
  101:
  102:     for (int x = 0; x < cap; x++) {
  103:         std::cout << buff[x] << " ";
  104:     }
  105:
  106:     std::cout << "\n\n";
  107: }
```