```
/******************************************************************************/
/* RingBuffer.cpp                                                             */
/* Yoo Min Cha                                                                */
/* RingBuffer                                                                 */
/* Professor Martin                                                           */
/* 16 March 2014                                                              */
/******************************************************************************/

#include "RingBuffer.hpp"

using namespace std;
using namespace sf;

RingBuffer::RingBuffer(int capacity):
ringBuff(capacity), _first(0), _last(capacity-1), _capacity(capacity), _full(false)
{
   if(capacity < 1)
          throw invalid_argument("Must be larger than zero");
}
int RingBuffer::size()
{
   return _capacity;
}
bool RingBuffer::isEmpty()
{
   if ( _first == 0 )
          return true;
   else
          return false;
}
bool RingBuffer::isFull()
{
   if (_first == _capacity)
          _full = true;
          return true;
   else
          _full = false;
          return false;
}
void RingBuffer::enqueue(Int16 x)
{
   if(isFull())
          throw runtime_error("Ring Buffer is full!");
   ringBuff[_first] = x;
   ++_first;
}
Int16 RingBuffer::dequeue()
{
   if(isEmpty())
          throw runtime_error("Ring Buffer is empty!");
   Int16 x = ringBuff[0];
   ringBuff.erase(ringBuff.begin(), ringBuff.begin()+1);
   ringBuff.push_back(0);
   --_first;
   return x;
}
Int16 RingBuffer::peek()
```

```
{
	if(isEmpty())
		throw runtime_error("Ring Buffer is empty!");
	return ringBuff[0];
}
```