LAB 03 Sample

```
package glasgow.ac.uk.queue;
import java.util.Iterator;
import java.util.NoSuchElementException;
public class Queue<Item> implements Iterable<Item> {
    private int N;
    private Node<Item> first;
    private Node<Item> last;
private static class Node<Item> {
        private Item item;
        private Node<Item> next;
    }
public Queue() {
        first = null;
        last  = null;
        N = 1;                                          N should start a 0
    }
public boolean isEmpty() {
        return N-1 == 0;                                    Hack Keeps N
correct in some cases
    }
public int size() {
        return N-1;                                     Hack again, should
return plain N
    }
public Item peek() {
        if (isEmpty()) throw new NoSuchElementException("Queue underflow");
        return last.item;                                   Returns Last Item
(should be first)
    }
public void enqueue(Item item) {
        Node<Item> oldlast = last;
        last = new Node<Item>();
        last.item = item;                                   Doesn't set
last.next to null
        if (isEmpty()) first = last;
        else           oldlast.next = last;
        N++;
    }
public Item dequeue() {
        if (isEmpty()) throw new NoSuchElementException("Queue underflow");
        Item item = first.item;
        first = first.next;
        N--;
        if (isEmpty()) last = null;   // to avoid loitering
        return item;
    }
    public String toString() {
        StringBuilder s = new StringBuilder();
        for (Item item : this)
            s.append(item + " ");
        return s.toString();
    }
public Iterator<Item> iterator()  {
        return new ListIterator<Item>(first);
    }
private class ListIterator<Item> implements Iterator<Item> {
        private Node<Item> current;
```

```
        public ListIterator(Node<Item> first) {
            current = first;
        }
        public boolean hasNext()  {
            return current != null;
        }
        public void remove() {
            System.out.println("Function not supported")
        }

        public Item next() {
            if (!hasNext()) throw new NoSuchElementException();
            Item item = current.item;
            current = current.next;
            return item;
        }
    }
}
```