# Multiple Templated Types

A template type for each Template parameter in the constructor was needed for the priority\_queue operators. Without it the template used the default types.

# Random Access Iterators

There is not an iterator defined for priority\_queue. This makes sense because it is an Adaptor type, but it meant that print was not easy to implement. You could define an iterator by inheriting from priority\_queue.

# Operator+

The operator+ poses an interesting problem here in that the two queues must have the same comparator. I have added a pre-condition that they be the same but I was unable to define a method that would add items with different comparators.

# Clear()

Does not exist for the priority\_queue. I found several opinions as to why, none of them seem to be that great. The general consensus is that you should inherit from the priority\_queue adaptor and do it yourself. I wrote a simple local function because that was easier.

# Items of Equal priority

Items of equal priority are not guaranteed to be returned in the order they are inserted. I verified through inspection that the priority queue is built on a heap. When the heap is re-heapified this can affect the ordering of items of an equal priority. Their order is actually dependent upon the height of the tree! This is also why the additions of PQ and PQ@ produce different results depending on the orders, because we are inserting one list into the other within +.

# Test Plan

Boolean operators of Priority String.

Operators + for PriorityQueue. Verify non-commutative.

Verify order theory.

Verify Max Heap operations

Test Output: I tried something different there this time, just run the Comparator.exe file to see my test output.