Priority Queue class prototypes.

Priority queues maintain collections of objects arranged for fast access to the least element.

Several prototype implementations of priority queues are supported.

XPPQs implement 2-ary heaps via XPlexes.

SplayPQs implement PQs via Sleater and Tarjan's (JACM 1985) splay trees. The algorithms use a version of ``simple top-down splaying'' (described on page 669 of the article). The simple-splay mechanism for priority queue functions is loosely based on the one used by D. Jones in the C splay tree functions available from volume 14 of the uunet.uu.net archives.

PHPQs implement pairing heaps (as described by Fredman and Sedgewick in *Algorithmica*, Vol 1, p111-129). Storage for heap elements is managed via an internal freelist technique. The constructor allows an initial capacity estimate for freelist space. The storage is automatically expanded if necessary to hold new items. The deletion technique is a fast ``lazy deletion'' strategy that marks items as deleted, without reclaiming space until the items come to the top of the heap.

All PQ classes support the following operations, for some PQ class **Heap**, instance **h**, **Pix ind**, and base class variable **x**.

h.empty() returns true if there are no elements in the PQ.

h.length() returns the number of elements in h.

ind = h.enq(x) Places x in the PQ, and returns its index.

x = h.deq()	Dequeues the minimum element of the PQ into x, or generates an error if the PQ is empty.
h.front()	returns a reference to the minimum element.
h.del_front()	deletes the minimum element.
h.clear();	deletes all elements from h;
h.contains(x)	returns true if x is in h.
h(ind)	returns a reference to the item indexed by ind.
ind = h.first()	returns the Pix of first item in the PQ or 0 if empty. This need not be the Pix of the least element.
h.next(ind)	advances ind to the Pix of next element, or 0 if there are no more.
ind = h.seek(x)	Sets ind to the Pix of x, or 0 if x is not in h.
h.del(ind)	deletes the item with Pix ind.