The implementation of my skiplist is basically normal, but in order to satisfy the extension requirements of the lab, my skiplist is somewhat out of the ordinary. The first is that my nodes (which hold events) Hold the elements to their right in an arraylist, rather than an array. I always start with the maximum level equal to 1, then double it and the size of head and tail whenever an element is added with a pillar height greater than maxLvl. In order to make the skiplist singly linked, nodes only know which elements are right of them, but this is not a problem because when I am searching for an element in the skiplist, I traverse right until the element to my right has value greater than what I am looking for, then move down and repeat until I see the value I desire. There is never any need to look backwards.

In order to account for the case where two events of the same year are present, my EventNodes are capable of storing multiple events; EventNodes contain an arraylist of events within which all events of their given year are stored. When inserting, I search the whole skiplist for a node with events of the same year as the one I am inserting, and if such a node is not present then I proceed with insertion as normal. If that node is present, then I check if the event is identical to one already stored in the node. Because the ".equals" method for events is not implemented, I check if the toString result for each node stored in the arraylist is identical that of the node I am inserting. If it is, then I return. If it the event is not already present, I add it to the internal arraylist of the EventNode and return.

In order to make findMostRecent function with only singly linked nodes, I simply store the current node and next node. I traverse the skiplist normally, moving down if I see a node larger than the one I am looking for, and once I have reached the bottom, I know that the current node is the most recent node prior to or at the given year.