Grace Portelance
Skip Lab
README.pdf

I implemented the skiplist by assigning each event to a node. Each node corresponded with a year, and held a pillar height and an array of events. They also have an ArrayList of next pointers.
Using this object I performed insert and remove as depicted in the class pseudocode. For insert, if a node was already present in the tree corresponding with the event I wished to insert, I simply added it to the array of Events in the node. Otherwise, I created a whole new node.

To findMostRecent, I had to consider two cases, if a node with that year already existed, and if that year did not exist in the tree. For the former, I simply checked to see if any element in a given node had the same year as the imput. If so, I returned the element array of that node. Otherwise, I traversed the entire list until I 'dropped off so to speak, at a point x. I returned that node's elements, x, as it was the closest year.

To implement findRange, I used a helper method called findSingleRecent. I used this to find a node x whose .next would be the node just before (or equal to) 'first'. This worked well, as the first command of findRange is that y=x.next.
Using this method, I began my search at the 0^{th} level at node x and simply did a horizontal walkthrough of that level (where every element is present) until I exceeded 'last'.

For the extensions, I decided to begin my implementation with only next pointers—I figured it would be easier to do that from the beginning than attempt to change the whole structure of my EventNode class. Not being able to call .prev meant that I had to save more sets of pointers, so, for example, I could set x.prev=y by doing y.next=x.

I implemented the resizing portion last, very similarly to the array resizing of the previous lab. I checked at the beginning of my insert method if the random height of a new node was > maxHeight, then created a new node with double the height of the previous. I set the new head pointers equal to all the existing head pointers, then added the rest as pointers to the tail. The one issue I had with this is that how I set up my code, I only create a new node after I am sure that a node with that year does not already exist. For my resizing, I choose a random r before checking, and use that to determine if I should double before I even decide if I'm creating a new node.
I determined that the consequences of this are pretty negligible, but it is worth considering that a little extra work could be removed if I restructured my insert method.