Programming implementation:
I modified Event class such that each Event has a height h and an array of Events representing the next event at each level.

In EventList class, I firstly declared integers negInf and posInf to represent the positive and negative infinity. Then, I assigned usedHeight to 1 since we are starting from 1 level and dynamically resizing the head and tail pillars. I also initiated the head and tail pointer (event) for each EventList.

In the **constructor** of EventList, I assigned the head and tail to have the keys (year) of negInf and posInf and description of null, since I would arrange the events from earliest (left of list) to latest (right of list). Then I assigned the next pointer of head at each level to tail (No need to use prev because I have modified for Signly-linked skip list).

In **insert(Event e),** I firstly generate a random height for e using the given random number generator, and then I create the next pointer array of size same as e's height. Next, I start from the head at the highest level of e, and insert e into each lower level right after the last event earlier than it by changing their pointers. In this way, all the event within the same year would be sorted as the order of them appeared in the input file.

In **remove(int year),** I started from the head at top level, and always try to find the last event before year at each level. Then I iterated through the events after this one until I find the first event that is not in this year, then I would make the last event before this year point to this first event after this year, thereby removing the list of events in this level. I went through all the levels.

In **findMostRecent(int year),** I started from head at the top level and find the last event before this **year**, just like a common find method of skip list does. Then, I recorded the number of this **year** that has at least one event and prior to this **year**, and I tried to find the last event before this prior year. I did this iteration again becase we are not sure if there is any event in this **year**, and if there is not, we would need to find the first event in the prior year, which is not simply **year** – 1. Then, I would use a counter to record how many events are in this found year, and return an array of events starting from the next event of the last event we found in the previous year. Also, we need to notice that there maybe no event happened either before or in this **year**, by which we would have prior year as negative infinity (head), and therefore we need to return null.

In **findRange(int first, int last),** I simply started from the top head, and iterate until I find the last event happened before **first** in the base level. Then I would keep going to the next event until I find the first event later then **last**, and use a counter to record how many events are there in between **first** and **last.** Finally, I would build an array from the next event of the last event before this **first** year

with length same as the value of my counter. If the counter turned out to be zero, I would return null.

Both our find methods would run in $\Theta(logn + m)$ time because we need to iterate through the returned array of length m. Insert would run in $\Theta(logn)$ because we used the similar method as common skip list. Although we used a common find method twice in Remove, it would still be running in $\Theta(logn)$ asymptotically.

Dynamic Resizing Head and Tail pillars:
Besides setting the usedHeight as 1 in the construction of EventList, I added a method updateHT to double the size of the pointer array head and the usedHeight. Everytime when inserting an event, I would keep updating the pillars until the usedHeight is larger than the event's height.

Singly-linked Skip List:
This is already demonstrated in the implementation explanantion. I kept the tails but have removed all the prev pointers, as all the events would only pointing to some events later then it, and the last event would point to the tail.