Problem2

For this problem we had to input stream of numbers in an int[] array
Data structure allows you to find kth largest element of array

```
   Void add(int[] a) {         // add function that adds an array "a"
to data structure

            node = head;     // adds pointer node to head and adds to
an array
            temp[] = head;      //temp storage created for value of
initial
            head =head.next; // pointer points to next
            add (next);         // add next

      }

      int getLargest(int k) {           // obtains kthsmallest value
and return it

         for (int k=0; k<array; k++)    // array loop to find largest
         {

            if (k > min)             // less than array size
               k= min.value            // finds the min value
associated from the main file calling from the function
               System.out.println (int[] arr)

               return k;
      }


   }
```

Implementation for the add(int[] a) achieve O(N) time because the add is of one element in the array to the data structure making it linear and stable.

For getLargest(int k) the implementation achieves O (klogN) time due to the stability of obtaining the kth smallest values and returns and array

O(N)and (klogN) are the time complexities in which O(N)is quicker

Constraints on add(int[a])running in O(N) time allows the value to stored in one at a time and the O(klogN) time for getLargest(int k)allows the stability of the sorted array implemented by the data structure