```java
class LinkedList <T> {

    public node head = null;

    class node {
        public double priority;
        T data;
        public node next;

        public node(int priority, T data){
            this.priority = priority;
            this.data = data;
        }
        public node(T data, double priority){
            this.priority = priority;
            this.data = data;
        }
    }

    /**
     * This method returns the first element in the list, simlar to "deleteMin"
     * @return first element data of type <T>.
     */
    public T poll(){
        T val;
        if (head==null){
            head=null;
            return null;
        }
        if (head.next==null){
            val = head.data;
            head=null;
            return val;
        }
        else {
            val = head.data;
            head=head.next;
            return val;
        }
    }

    /**
     * This method inserts values into a priority
```

```
43 queue and sorts by the priority Max -> Min
44      * Trims list if size is greater than 20. used
   specfically for TOP 20 method in main.
45      * @param data Sorts stored data of specificed
   type T
46      * @param priority Sorts stored priority of type
   int by big -> small
47      */
48     public void insert(T data, int priority) {
49         node nodeToAdd = new node (priority,data);
50 
51         node current = head;
52         node temp;
53 
54         if (head == null) {
55             head = nodeToAdd;
56             return;
57         }
58         else if (nodeToAdd.priority > head.priority){
59             head = nodeToAdd;
60             head.next = current;
61             return;
62         }
63         else{
64             while (current != null) {
65 
66                 if (current.next!=null && current.
   next.priority < nodeToAdd.priority){
67                     temp = current.next;
68                     current.next = nodeToAdd;
69                     nodeToAdd.next = temp;
70                     break;
71                 }
72                 if (current.next == null) {
73                     current.next = nodeToAdd;
74                     break;
75                 }
76                 current = current.next;
77             }
78         }
79         if (getLength() > 20){
80             current = head;
81             for (int i=0;i<19;i++){
82                 current = current.next;
```

```
 83              }
 84              current.next = null;
 85          }
 86      }
 87
 88      /**
 89       * This method inserts values into a priority
    queue and sorts by the priority Min -> Max
 90       * Trims list if size is greater than 20. used
    specfically for TOP 20 method in main.
 91       * @param data Sorts stored data of specificed
    type T
 92       * @param priority Sorts stored priority of type
     int by small -> big
 93       */
 94      public void add(T data, double priority) {
 95          node nodeToAdd = new node (data,priority);
 96
 97          node current = head;
 98          node temp;
 99
100          if (head == null) {
101              head = nodeToAdd;
102              return;
103          }
104          else if (nodeToAdd.priority < head.priority
    ){
105              head = nodeToAdd;
106              head.next = current;
107              return;
108          }
109          else{
110              while (current != null) {
111                  if (current.next!=null && current.
    next.priority > nodeToAdd.priority){
112                      temp = current.next;
113                      current.next = nodeToAdd;
114                      nodeToAdd.next = temp;
115                      break;
116                  }
117                  if (current.next == null) {
118                      current.next = nodeToAdd;
119                      break;
120                  }
```

```
121                     current = current.next;
122                 }
123             }
124         }
125         public double getLargestModule(){
126             return head.priority;
127         }
128 
129 
130         public int getLength() {
131             int c = 0;
132             node current = head;
133             while (current != null) {
134                 c++;
135                 current = current.next;
136             }
137             return c;
138         }
139         public void printSubTreeSizes(){
140             node current = head;
141             while (current!=null){
142                 System.out.print((int)current.priority+
    " ");
143                 current = current.next;
144             }
145         }
146         public boolean isEmpty(){
147             if (head==null){
148                 return true;
149             }
150             else{
151                 return false;
152             }
153         }
154 }
155 
```