Problem 1:

To implement this problem, I used a custom concurrent linked list that stored a thank you card count and size, the linked list was fully locked whenever an access was made ensuring mutual exclusion. Involved nodes and node operations in addition to all other standard necessities of a linked list. The Servant class extended the java thread class and allowed it to choose a random function to call of the three necessary functions: search, insert and delete. The mutual exclusion of the linked list and 500,000 thank you cards being sent are proof of correctness.

```
499939: in List
499559: in List
499960: in List
499974: in List
499841: in List
499710: in List
499777: in List
499712: in List
499729: in List
499705: in List
489216: in List
490128: in List
499749: in List
461814: in List
495824: in List
474050: in List
499805: in List
496900: in List
499854: in List
499857: in List
499771: in List
499451: in List
499738: in List
499859: in List
499312: in List
499335: in List
499932: in List
499901: in List
499921: in List
499835: in List
499833: in List
499873: in List
499928: in List
493522: in List
499786: in List
483720: in List
499944: in List
499786: in List
481420: in List
499845: in List
494195: in List
485744: in List
498943: in List
493159: in List
499016: in List
498455: in List
496027: in List
499886: in List
496431: in List
Thank you cards sent: 500000
```

For problem 2 I used a shared array list and gave each thread a range of indices to manage. The shared array list did not require mutual exclusion as the indices only belonged to their respective thread and were indexed by tick.

Then there were two smaller algorithms. One stored the sorted list in a stream and built another array to see the top temperatures and lowest temperatures. This is in addition to the other algorithm that found the max temperature in each ten-minute interval and the

main temperature in each ten-minute interval and calculated the skew, it then compared the skews to find the maximum temperature skew.

This approach is correct because it ensures that each range of indexes in shared memory can be managed by its own thread and as it does not rely on any data from another thread there are no memory hazards.

```
Lowest Temps: -100 -100 -100 -99 -99
Highest Temps: 68 68 68 69 69
Max skew Interval: 5
```