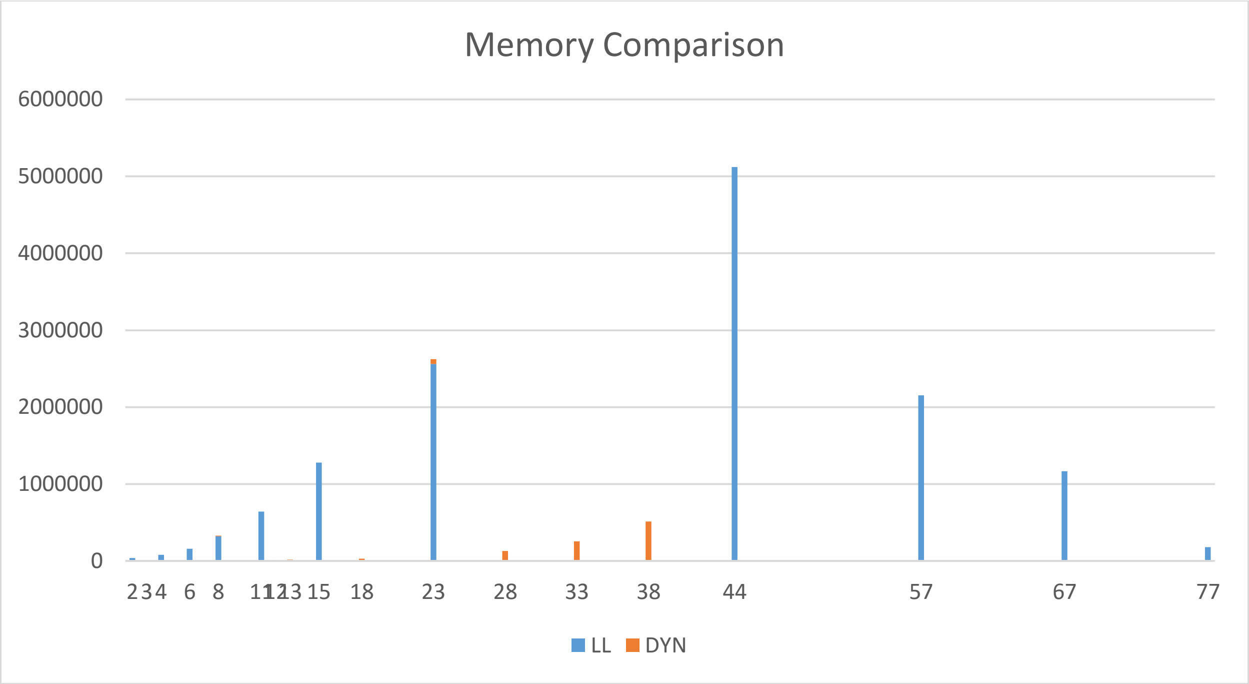
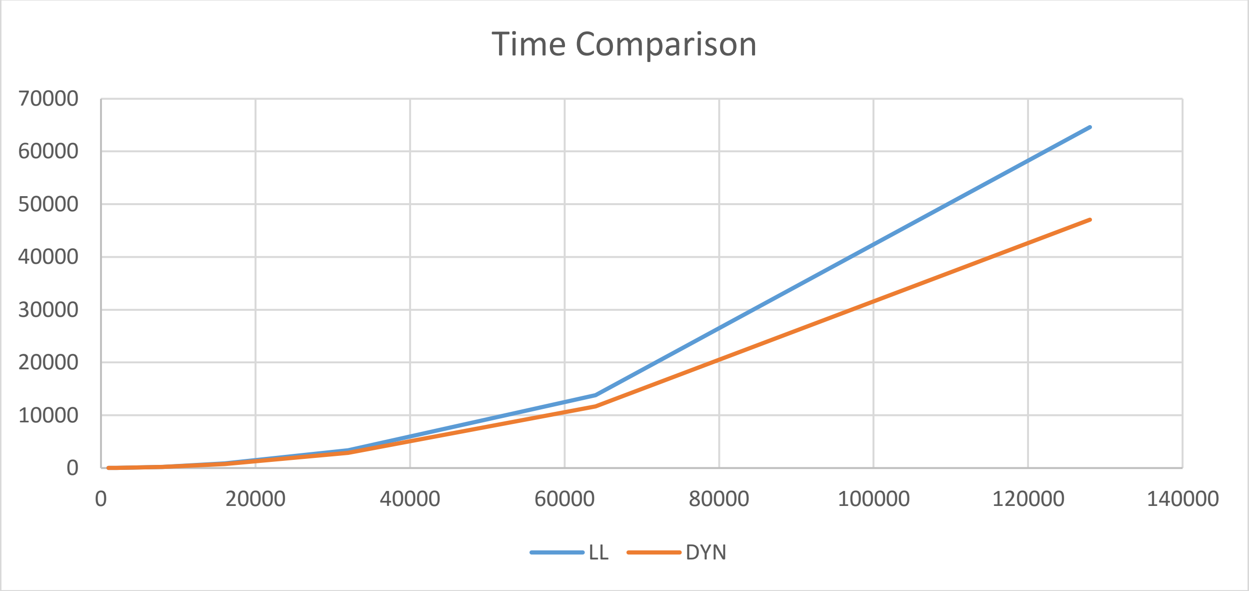

Memory Comparison
6000000
5000000
4000000
3000000
2000000
1000000
0
2
3
4
6
8
11
12
13
15
18
23
28
33
38
44
57
67
77
LL
DYN


Time Comparison
70000
60000
50000
40000
30000
20000
10000
0
0
20000
40000
60000
80000
100000
120000
140000
LL
DYN

1. **Which of the implementations uses more memory? Explain why.**
   Overall the linked list uses more memory. The reason for this is because when you allocate a chunk of memory there is some baggage that comes with it. The linked list has more baggage because for every link that gets allocated there is some extra space needed. The dynamic array, no matter how much memory is allocated, only needs extra for that one allocation. Essentially, you will get more efficiency in memory allocation by using a dynamic array rather than a linked list.

2. **Which of the implementations is the fastest? Explain why.**
   The fastest implementation is the dynamic array. The reason for this is because you don’t have to jump from link to link. Simply, the computer can perform a binary search on the dynamic array because the memory is contiguous.

3. **Would you expect anything to change if the loop performed remove() instead of contains()? If so, what? (Note, it's very easy to run this experiment given the code we've provided!)**
   The linked list would be faster in this regard. The reason is because there is less operational work than with the dynamic array. In a linked list implementation, Remove() would simply find the element, then point neighboring elements to one another. In a dynamic array implementation, Remove() would find the element (faster), but then it would have to shift every single element in the array forward or back by one element and this is very costly in terms of time.