1. Who is your programming partner? Which of you submitted the source code of your program?
   My programming partner was Aaron smith, I submitted our source code.
2. Evaluate your programming partner. Do you plan to work with this person again?
   My partner helped me a lot with this project and I plan on working with him again.
3. Collect and plot running times in order to answer each of the following questions. Note that this is this first assignment that does not specify the exact procedure for creating plots. You must design your own timing experiments that sufficiently analyze the problems. Be sure to explain all plots and answers.
   - Is the running time of the `addFirst` method *O*(1) as expected? How does the running time of `addFirst(item)` for `MyLinkedList` compare to `add(0, item)` for `ArrayList`?

     Our running time for addFirst plots a line that doesn't seem to be increasing, this means that it behaved as expected and is constant, or O(1). When comparing Addfirst with add(0) for an arrayList our linked list performs much better. This is because an arraylist must iterate through the entire list every time it adds to the beginning, giving it linear growth, where as in a linked list you only need to modify the head.

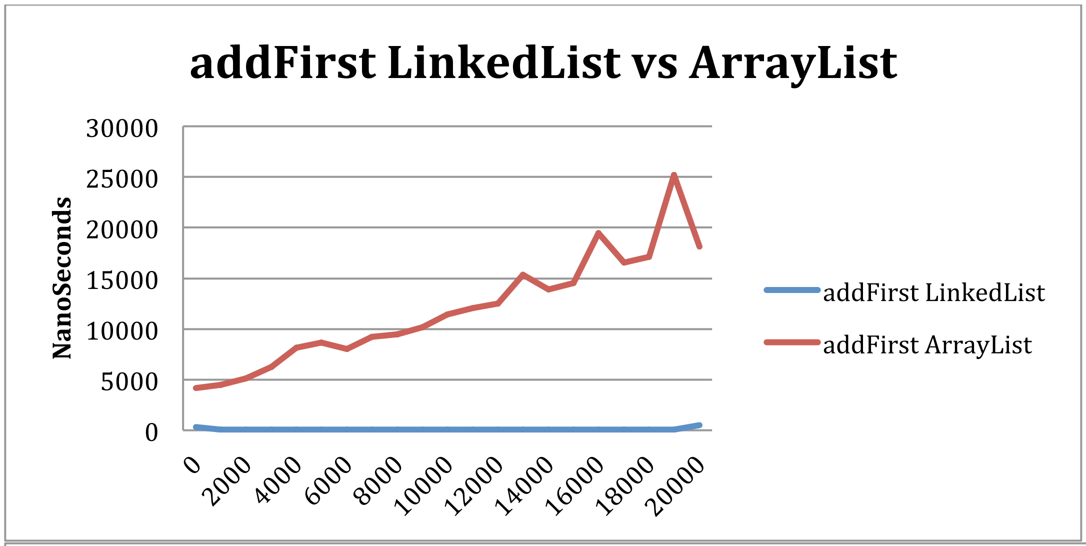
   - Is the running time of the `get` method *O*(*N*) as expected? How does the running time of `get(i)` for `MyLinkedList`compare to `get(i)` for `ArrayList`?

     When comparing our get method with the arrays get method the array list preforms much better. It is almost the exact opposite case as described above. Our linked list must iterated through the list in order to search for the right index where as the array list can assume the memory location and directly access it. Because of this our get method as O(N) behavior where the array list has a constant O(1) growth.

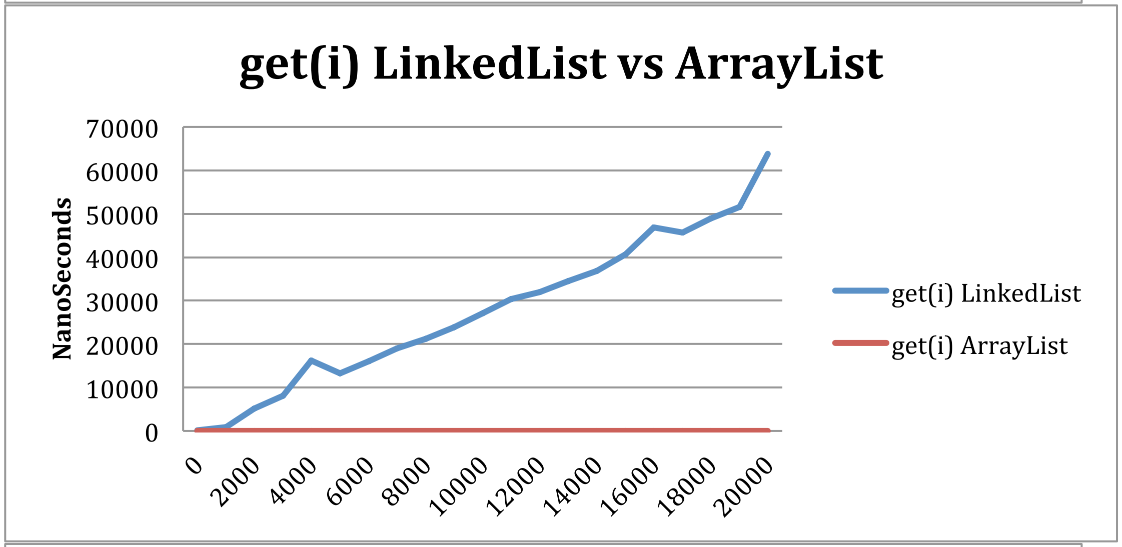
   - Is the running time of the `remove` method *O*(*N*) as expected? How does the running time of `remove(i)` for`MyLinkedList` compare to `remove(i)` for `ArrayList`?

     For the remove method in both the linked list and the array list, they both appear to have O(N) growth. This is because in the linked list it must iterate through the entire list to find where to remove, and in the array list it must move all the items to the right left one space.

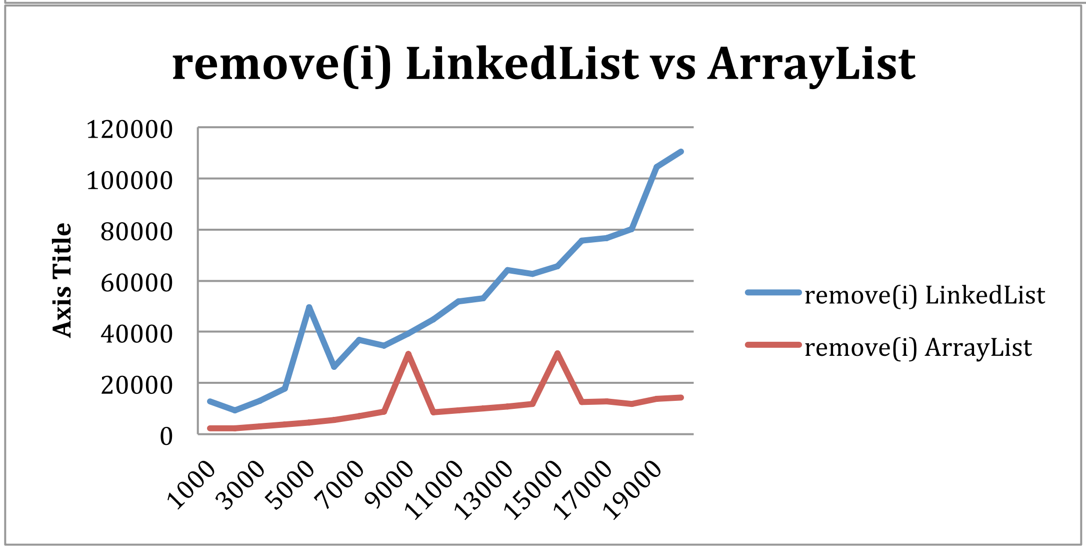
4. In general, how does `MyLinkedList` compare to `ArrayList`, both in functionality and performance?
   In general Linked List are better for performance if you are just adding or removing items from the beginning or end of the array, such as a queue list, because you don't have to modify or iterate through any other items besides the head or tail. Array List are bad at adding items at the beginning or end of the list because it must move all items to the right to add to the beginning and it must also grow the array if the array is too small. Array List out perform linked list though on getting items from it. This is because an array list can assume the memory location of its items so it can directly jump to the index of the item needed without iterating through every item, unlike an linked list.
5. In general, how does `MyLinkedList` compare to Java's `LinkedList`, both in functionality and performance?
   In general both of the linked list appear to have the same growth rates, meaning that they are using the same method to get and remove items. Java's linked list though appears to have less spikes in it performance, this might be due to some optimizations in its code that we did not practice.
6. How many hours did you spend on this assignment? We spent a total of 6 hours on this assignment.



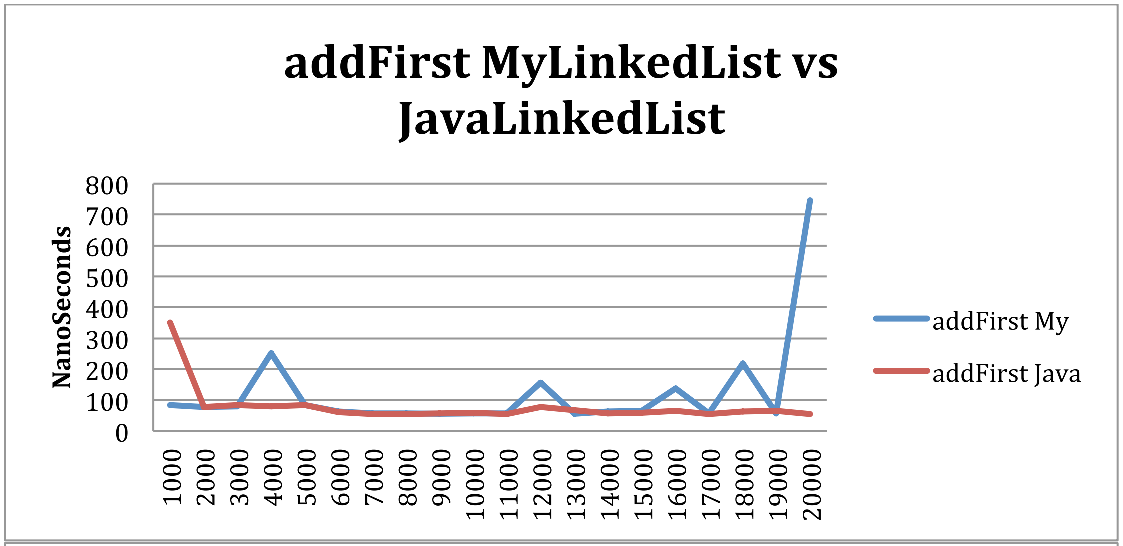
addFirst LinkedList vs ArrayList
NanoSeconds
30000
25000
20000
15000
10000
5000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
addFirst LinkedList
addFirst ArrayList



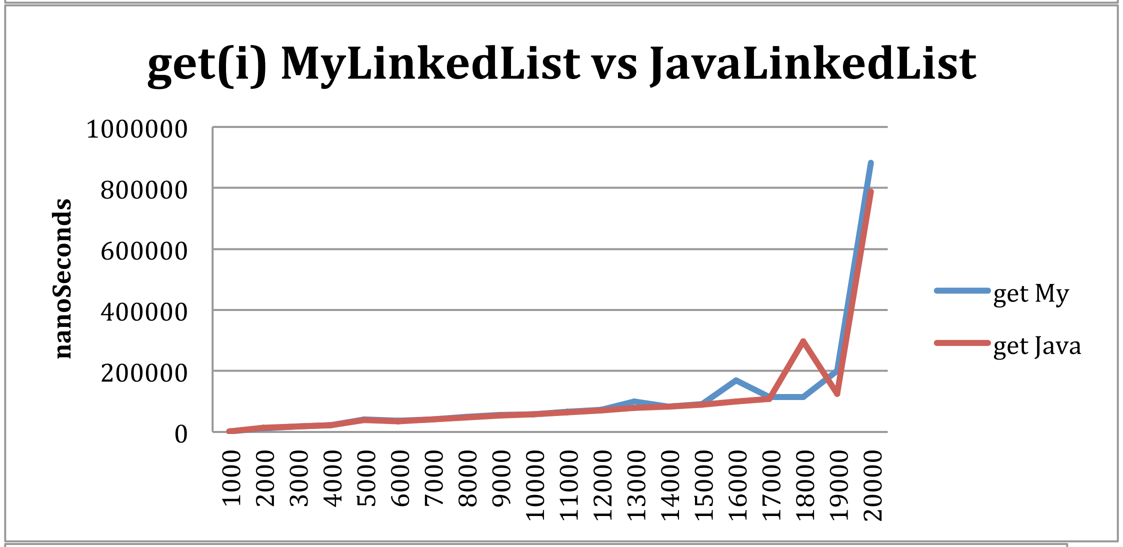
get(i) LinkedList vs ArrayList
NanoSeconds
70000
60000
50000
40000
30000
20000
10000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
get(i) LinkedList
get(i) ArrayList



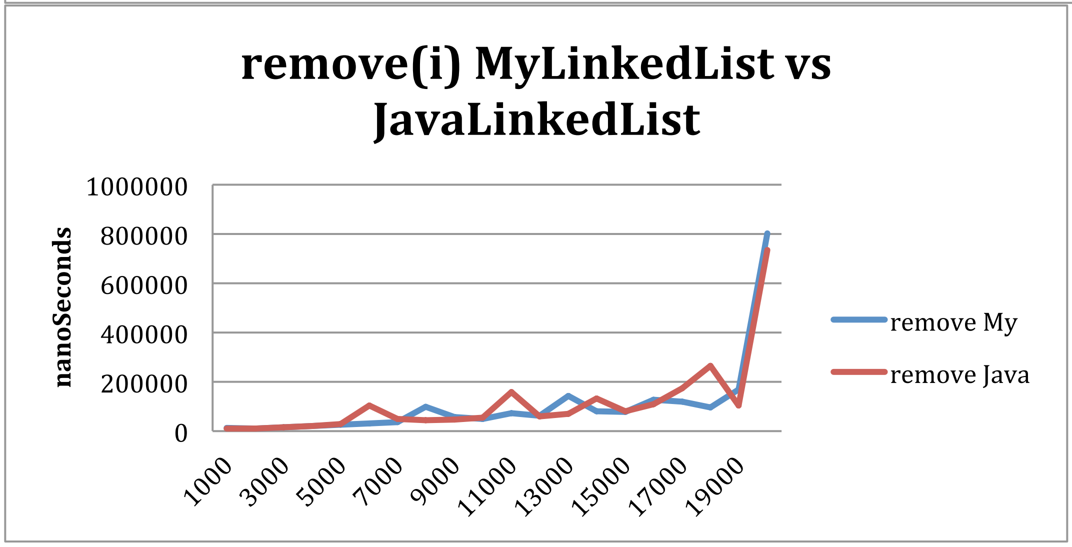
remove(i) LinkedList vs ArrayList
Axis Title
120000
100000
80000
60000
40000
20000
0
1000
3000
5000
7000
9000
11000
13000
15000
17000
19000
remove(i) LinkedList
remove(i) ArrayList



addFirst MyLinkedList vs JavaLinkedList
NanoSeconds
0
100
200
300
400
500
600
700
800
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
addFirst My
addFirst Java



get(i) MyLinkedList vs JavaLinkedList
nanoSeconds
0
200000
400000
600000
800000
1000000
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
get My
get Java



remove(i) MyLinkedList vs JavaLinkedList
nanoSeconds
0
200000
400000
600000
800000
1000000
1000
3000
5000
7000
9000
11000
13000
15000
17000
19000
remove My
remove Java