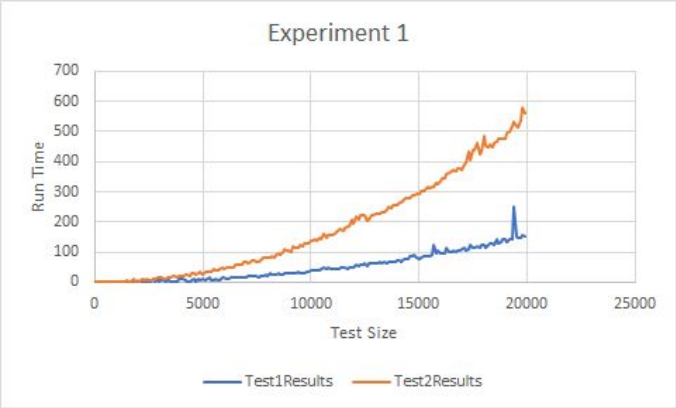
1.

In DoubleLinedList, since only one Indexof() method navigate through list based on based on the actual data of the list node, the rest of method navigate through list only based on index, we only need to consider null case in indexOf() method. So we add a condition when decide whether to return node (item == null && cur.data == null) to hand the data=null case.

In ArrayDictionary, since the key of pair can be null, we add special case when we need to get the key value and compare to required value. (key == null && pairs[i].key == null)

2.

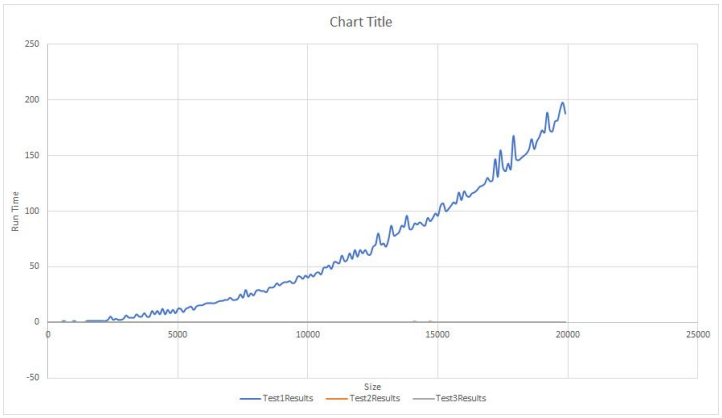
Ex1:

1) Ex1 tests the efficiency of remove method for ArrayDictionary both from front and from back and write the result to a csv file
2) Remove method run time is O(log(n)) since n get smaller when remove call repeatedly. Since there is about 200 object in ArrayDictionary, the run time could be around log(size of that test run) * 10^-6 second for each test
3) 

4) Test 2 is a log() function which is same to our prediction, test 1 is about constant time. The difference we believe is because test 1 remove from front which has constant run time and test 2 run from back which has log(O) run time

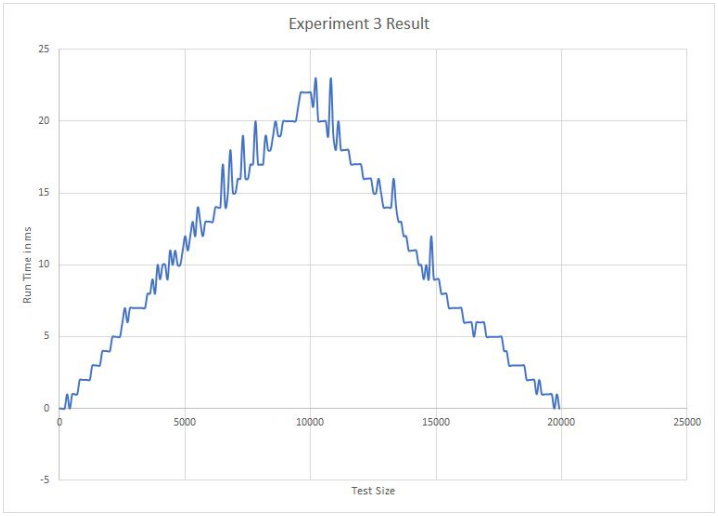
Ex2:

1) Ex2 loop through each value in the doublelinkedlist using three different method: for loop and get method, for each loop and iterator
2) For loop and get : Log(O) run time
   For each loop: N run time
   Iterator: N run time

3) 

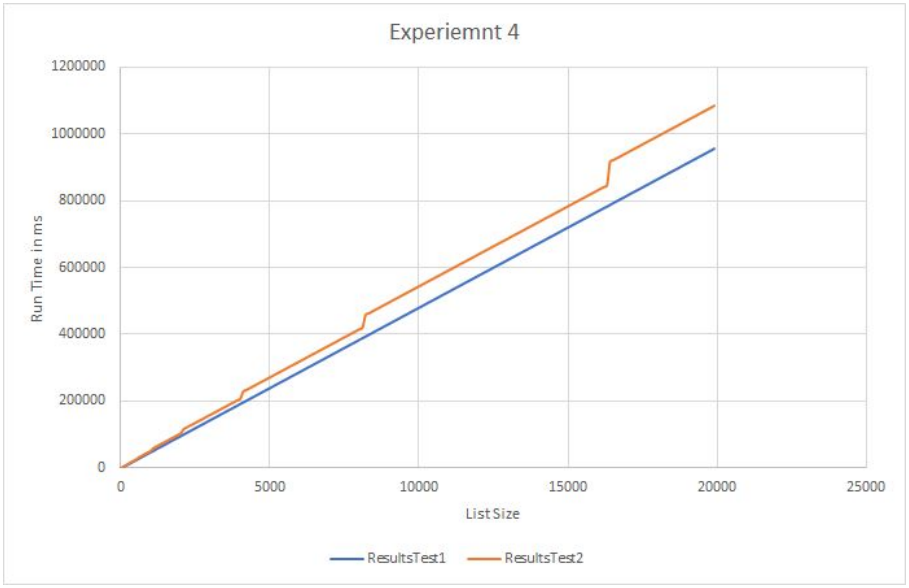
4) Since test 1 has get method which also has its own for loop, the run time of that will be O(N^2) while the other two test only has to go over list once.

Ex3

1) Ex3 test the efficiency of get method
2) The run time will be N/2 since we implement a faster algorithm that if the index is in last half of the list, we loop out list from behind
3) 

4) As expected, we implement a faster algorithm that if the index is in last half of the list, we loop out list from behind

Ex 4

1) Chech how many memory is used to construct a list of N element
2) Liner since we only need add element to the end once at constant run time
3) 

4) As expected, since we only need add element to the end once at constant run time