[**Closed book and notes.**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided.

1. (7 points) Recall that the asymptotic time complexity for our implementation of the add(E element) method from the LinkedList class was $O(n)$ whereas the JCF implementation is $O(1)$. Explain why these differ.

2. (7 points) Recall the ArrayList class that we developed in lecture. Give the big-oh time complexity for the implementation of the add(E element) we did in class. Justify your answer.

3. (6 points) Suppose that the LinkedList implementation we did in class was modified so that the head attribute was replaced with a tail attribute that pointed to the last element in the list. Furthermore, the inner class is modified to have a prev attribute instead of a next. Give the big-oh time complexity for the most efficient implementation of add(E element) that you can think of. Justify your answer.

4. (15 points) Recall that the ArrayList class that we developed in lecture has one attribute: data that is an array containing all of the elements in the ArrayList. Implement the remove(Object element) method. (This was a homework problem.)

5. (15 points) Recall that our implementation of the LinkedList<E> from lecture contains one attribute: head which is a reference to the first Node in the list. Implement the add(int index, E element) method. (This was a homework problem.)

6. (20 points) Implement an inner class for the LinkedList we developed in lecture that implements the Iterator<E> interface. You do not need to provide an implementation for the remove method. (This was a homework problem.)

7. Consider the following method:

```
public static double getMedian(List<Double> nums)
{
  int N = nums.size();
  if(N<1)
  {
    throw new Exception("Grip a get");
  }
  double median = nums.get(0);
  int index = 0;
  for(int i=0; i<N/2; ++i)
  {
    int j;
    for(j=1; j<N-i; ++j)
    {
      if(nums.get(j)<median)
      {
        median = nums.get(j);
        index = j;
      }
    }
    swap(nums, j-1, index);
  }
  return median;
}
```

(a) (10 points) Suppose that the List passed to the method is a java.util.ArrayList and assume that swap swaps the element at $(j-1)$ and element at $index$ in nums and runs in $O(N^2)$ time. Using big-oh notation, describe the overall worst case time complexity for the getMedian method. Be sure to explain your reasoning and state any additional assumptions that you make.

(b) (10 points) Suppose that the List passed to the method is a java.util.LinkedList and assume that swap swaps the element at $(j-1)$ and element at $index$ in nums and runs in $O(N)$ time. Using big-oh notation, describe the overall worst case time complexity for the getMedian method. Be sure to explain your reasoning and state any additional assumptions that you make.

(c) (10 points) Suppose that the List passed to the method is the LinkedList developed in lecture and assume that swap swaps the element at $(j-1)$ and element at $index$ in nums and runs in $O(1)$ time. Using big-oh notation, describe the overall worst case time complexity for the getMedian method. Be sure to explain your reasoning and state any additional assumptions that you make.