Assignment 3

A. Design a pseudo code algorithm to take a Sequence and remove all duplicate elements from the Sequence. Is the algorithm the same for both a List or a Sequence? Explain. Analyze your algorithm twice, once assuming it is a Sequence and once assuming it is a List. Which ADT is a better choice for this problem? Implement your choice in JavaScript.

```
Algorithm: removeDuplicate(L)
        p←L.first()
while !L.isLast(p) do                          --------- n
        e←L.element()                          --------- n
        removeDuplicateHelper(e, L.after(p), L)  --------- n^2

Algorithm: removeDuplicateHelper(e, p, L)
        If L.isLast(p) then
            If e==plelment() then
                L.remove(p)
        Else
        q ← L.after(p)
                if e==p.element() then
                   L.remove(p)
        removeDuplicateHelper(e, q, L)
```

using Sequence

```
Algorithm: removeDuplicate()
                if (this.isEmpty())
                throw new Error("sequence isempty "); }
                else
                for let i ← 0; i  to this.size() - 1
                    for (let j ← i + 1; j  to  this.size()
                        if (this.elemAtRank(i) == this.elemAtRank(j))
                        this.removeAtRank(j);
                        else
                        return this
```

B. Design an algorithm, isPermutation(A, B) that takes two sequences A and B and determines whether or not they are permutations of each other, i.e., same elements but possibly occurring in a different order. Hint: A and B may contain duplicates, thus if A contains three x's, then B must also contain exactly three x's. What is the worst-case time complexity of your algorithm? Justify your answer. Implement your algorithm in JavaScript using either the Sequence or the List program provided.

Algorithm: ispermutation(A, B)

```
If  A.size() !== B.size()                                   --------O(1)
        return not permutation                              ---------O(1)
Else
        P ← A.sort()                                        ------- O(n)
        Q ← B.sort()                                        --------O(n)
        For i ←  0 to A.size()                              --------O(n)

                If P.elementAtRank(i) !== Q.elementAtRank(i)    --------O(n)
                        Return not permutation                  --------O(1)
                Else
                        Return it is permutation                ---------O(1)
```

So running time is ------**O(n)**