**Assignment 12**

**C-4.11 Suppose we are given an *n*-element sequence S such that each element in S  
represents a different vote in an election, where each vote is given as an integer  
representing the ID of the chosen candidate. Suppose we know who the candidates are  
and the number of candidates running is *k* < *n*. Describe an *O(n* log *k)*-time pseudo code  
algorithm for determining who wins the election.**

1. **Suppose we are given an *n*-element list S such that each element in S  
   represents a different vote in an election, and where each vote is given as an  
   integer representing the ID of the chosen candidate. Describe an O(n) pseudo  
   code algorithm for determining who wins the election (turn in a pseudo-code  
   version as well). This clearly needs to use a hash table based Dictionary, i.e., the  
   HT\_Dictionary class. Your code goes in the file HW12\_VotingHelpers.js. There is  
   a comment in the functions that you are to write that states what the function is to  
   do; these are building blocks showing different ways of solving this problem, e.g.,  
   differing inputs (array vs. Sequence) and different approaches, e.g., using a  
   Dictionary. For example, LookupTable (LookupTable.js), unordered dictionary  
   (HT\_Dictionary.js), Priority Queue (PriorityQueue.js).**

Algorithm voteCount(S, D)

Iter := S.elements()

while iter.hasNext() do

e := iter.nextObject()

c := D.findValue(e)

if c = null then

D.insertItem(e, 1)

else

D.insertItem(e, c+1)

countElementsOfSeq(seq, D) {

        // count the elements in seq and store the count for each candidate in Dictionary D

        let p = seq.first();

        D.insertItem(p.element(), 1);

        while (!(seq.isLast(p))) {

            p = seq.after(p);

            let c = D.findValue(p.element());

            if (c === null) {

                D.insertItem(p.element(), 1);

            } else {

                D.insertItem(p.element(), c + 1)

            }

        }

    }

countElementsOfArray(arr, D) {

        // count the elements in seq and store the count for each candidate in Dictionary D

        for (let i = 0; i < arr.length; i++) {

            let e = arr[i];

            let c = D.findValue(e)

            if (c === null) {

                D.insertItem(e, 1)

            } else {

                D.insertItem(e, c + 1);

            }

        }

    }

\_findWinnersFromDictionary(D) {

        // The count for each candidate should be in Dictionary D

        // Iterate through the Items (ID, count) and find the winners and put in the array

        let iterD = D.items();

        let winners = [];

        let max = 0;

        while (iterD.hasNext()) {

            let item = iterD.nextObject();

            let key = item.key();

            let c = D.findValue(key);

            if (c > max) {

                max = c;

                winners = [];

                winners.push(item);

            } else if (c === max) {

                winners.push(item);

            }

        }

        return winners;

    }

    \_insertSeqIntoPQ(seq, PQ) {

        // insert the elements (candidate ID's) from Sequence seq into the Priority Queue PQ

        let iter = seq.elements();

        // your code goes here

        while (iter.hasNext()) {

            let e = iter.nextObject();

            PQ.insertItem(e, e);

        }

    }

    \_insertArrayIntoPQ(arr, PQ) {

        // insert the elements (candidate ID's) from Sequence seq into the Priority Queue PQ

        for (let i = 0; i < arr.length; i++) {

            let e = arr[i];

            PQ.insertItem(e, e);

        }

    }

    \_findWinnersFromPQ(PQ) {

        // Traverse the Priority Queue and determine the winners

        let max = 0;

        let winners = [];

        let curr = PQ.removeMin();

        let c = 1;

        while (!PQ.isEmpty()) {

            let next = PQ.removeMin();

            if (curr === next) {

                c ++;

            } else {

                if (c > max) {

                    winners = [];

                    winners.push([curr, c]);

                    max = c;

                } else if (c === max) {

                    winners.push(([curr, c]));

                }

                curr = next;

                c = 1;

            }

        }

        if (c > max) {

            winners = [];

            winners.push((curr, c));

            max = c;

        } else if (c === max) {

            winners.push((curr, c));

        }

        return winners;

    }