```
const { List } = require("./List");
function listUnion(A, B) {
  let union = new List();
  let a = A.first();
  let b = B.first();
  if (a.element() > b.element())
union.insertLast(b.element());
  else union.insertLast(a.element());
  while (!(A.isEmpty() || B.isEmpty())) {
    a = A.first();
    b = B.first();
    if (a.element() === b.element()) {
      if (!contains(union, a.element(), union.first())) {
        union.insertLast(a.element());
      }
      A.remove(a);
      B.remove(b);
    } else if (a.element() < b.element()) {
      if (!contains(union, a.element(), union.first())) {
        union.insertLast(a.element());
      }
      a = A.remove(a);
    } else {
      if (!contains(union, b.element(), union.first())) {
        union.insertLast(b.element());
      }
      b = B.remove(b);
    }
  }
  while (!A.isEmpty()) {
```

```
    a = A.first();
    if (!contains(union, a.element(), union.first())) {
      union.insertLast(a.element());
    }
    a = A.remove(a);
  }
  while (!B.isEmpty()) {
    b = B.first();
    if (!contains(union, b.element(), union.first())) {
      union.insertLast(b.element());
    }
    b = B.remove(b);
  }
  return union;
}

function contains(list, e, p) {
  if (list.isEmpty()) return false;
  if (e === p.element()) return true; //n
  if (list.isLast(p)) return false;
  return contains(list, e, list.after(p)); //n
}
```

Algorthim SortRBG(seq)
  pq=new PriortyQue()
pQColorSort(seq,pq)

Algorthim pQColorSort(seq,pq)
    While seq.size()>0 O(n)

```
        e:=seq.remove(seq.first()).          O(n)
      if(e===”Reda”)                         O(n)
       pq.inserItem(1,e)                      O(nlogn)
   else if(e===”Blue”)                        O(n)
      pq.insertItem(2.e)                     O(nlogn)
else if (e===”Green”).                        O(n)
   pq.insertItem(3,e)                       O(nlogn)

while(pq.size()>0)                           O(n)
e:=pq.removeMin()                           O(nlogn)
  seq.insertLast(e)                         O(n)
return seq;                                 O(1)
```

Big O of this is O(NlogN)

```
SortRBG (seq){
         let PQ=new pq.PriorityQueue();
         this._PQColorSort(seq,PQ)
}
     _PQColorSort(seq,PQ){
        while(seq.size()>0) {
            let e=seq.remove(seq.first())
             if(e==="Red")
              PQ.insertItem(1,e)
            else if(e==="Blue")
              PQ.insertItem(2,e)
           else if(e==="Green")
             PQ.insertItem(3,e)
```

```
        }
        while(PQ.size()>0){
          let e=PQ.removeMin()
           seq.insertLast(e)
        }
    }
```

```
const {Sequence} = require('./Sequence.js');
function electionResult(seq){
let v=seq.first()
let count=0;
let eachCount=0;
let winner=v;
while(!seq.isLast(v)){
     eachCount=countVote(seq,v)
    if(eachCount>count){
        count=eachCount;
        winner=v
     }
    v=seq.after(v)
 }
 eachCount=countVote(seq,v)//last candidate
 if(eachCount>count){
    count=eachCount;
    winner=v
  }
  return winner;
}
 function countVote(seq,v){
    let p=seq.first();
    let count=0;
```

```
    while(!seq.isLast(p)){
        if(v.element()===p.element())
              count++;
        p=seq.after(p)
    }
    if(v.element()===p.element())
            count++;
```

```
const {Sequence} = require('./Sequence.js');
function electionResult(seq){
let v=seq.first()
let count=0;
let eachCount=0;
let secondCount=0
let firstWinner=v;
let secondWinner=v;
let result=[];
while(!seq.isLast(v)){
     eachCount=countVote(seq,v)
     if(eachCount!==count){
         if(eachCount>count){
            secondCount=count;
            secondWinner=firstWinner;
            count=eachCount;
            firstWinner=v
         }else if(eachCount>secondCount){
            secondCount=eachCount;
            secondWinner=v;
         }
```

```
    }
    v=seq.after(v)
 }
 eachCount=countVote(seq,v)
 if(eachCount!==count){
    if(eachCount>count){
        count=eachCount;
        firstWinner=v
      }else if(eachCount>secondCount){
        secondCount=eachCount;
        secondWinner=v;
    }
 }
 result=[];
  result.push(firstWinner)
  result.push(secondWinner)
 return result;
}
 function countVote(seq,v){
    let p=seq.first();
    let count=0;
    while(!seq.isLast(p)){
        if(v.element()===p.element())
              count++;
        p=seq.after(p)
    }
    if(v.element()===p.element())
            count++;
 return count;
 }
```

```
const {Sequence} = require('./Sequence.js');
function electionResult(seq){
let v=seq.first()//1
let count=0;//1
let eachCount=0;//1
let secondCount=0//1
let firstWinner=v;//1
let secondWinner=v;//
let result=[];
while(!seq.isLast(v)){//n
     eachCount=countVote(seq,v)//n*n
         if(eachCount>count){//n/2
            secondCount=count;//n/2
            secondWinner=firstWinner;//n/2
            count=eachCount;//n/2
            firstWinner=v//n/2
         }else if(eachCount>secondCount){//n/2
            secondCount=eachCount;//n/2
            secondWinner=v;//n/2
         }
    v=seq.after(v)//n
 }
 eachCount=countVote(seq,v)//1
    if(eachCount>count){//1
      secondCount=count;
      secondWinner=firstWinner;
      count=eachCount;
      firstWinner=v
   }else if(eachCount>secondCount){
      secondCount=eachCount;
      secondWinner=v;
   }
```

```
  result.push(firstWinner)//1
  result.push(secondWinner)//1
 return result;//1
}
//O(n^2)
 function countVote(seq,v){//
    let p=seq.first();//1
    let count=0;//1
    while(!seq.isLast(p)){//n
        if(v.element()===p.element())//n
              count++; //n
        p=seq.after(p)//n
    }
    if(v.element()===p.element())//1
            count++; //1
 return count;//1
 }
 let list1 = new Sequence();//list of candidates and
number of votes
```