**(Main)**

//parsing code omitted

Create new que for Democrats (queDem)

Create new que for Republicans (queRep)

Create new que for Whigs (queWhig)

While (stack is not empty)

Create temp and set to stack pop

If (temp equals “Democrat”)

Insert temp into queDem

Else if (temp equals “Republican”)

Insert temp into queRep

Else if (temp equals “Whigs”

Insert temp into queWhigs

Else

Return “Bad Data”

//end While

//for Updated info

While (stack is not empty)

Create temp and set to stack pop

If (temp’s getParty equals “Democrat”)

Insert temp into queDem

Else if (temp equals “Republican”)

Insert temp into queRep

Else if (temp equals “Whigs”

Insert temp into queWhigs

Else

Return “Bad Data”

//end While

//end Main

//end Main Class

**(Queue)**

<Constructor> Create array list queArray of type Presidents

Set front to 0

Set rear to -1

Set nItems to 0

//end constructor

Insert(Presidents j)

Add j to queArray

Increment rear

Increment nItems

//end insert

Delete()

Set temp to item removed from queArray

Decrement rear

Decrement nItems

Return temp

//end Delete

Display()

for(set i to 0, if I less than or equal to rear, increment i)

get i from queArray and assign to temp

//end Display

peekFront ()

return front

//end peekFront

isEmpty()

if nItems is 0, then queue is empty

//end isEmpty

Sort()

for( set j to 0, if j is less than rear, increment j)

for( set i to j+1, if i is less than rear+1, increment i)

if(queArrays’ i is less than queArrays’ iMin)

Set i to iMin

If(iMin doesn’t equal j)

swap

//end Sort

swap(int j, int iMin)

set temp to iMin removed from queArray

add(j, temp)

//end swap

insertCorrect (President p)

for(set i to 0, if i is less than or equal to nItems, increment i)

if ( p’s number is less than queArray’s i and i equals 0)

add(i,p)

increment nItems

increment rear

else if (p’s number is less than queArray’s i)

add((i)p)

increment nItems

increment rear

else if ( i equals rear)

add(p)

increment nItems

increment rear

//end insertCorrect

//end Queue Class

**(StackP)**

Declare top, bottom and nItems2 as integers

<Constructor> StackP()

Create ArrayList of President objects

Set top to 0

Set bottom to -1

Set nItem2 to 0

//end Constructor

push(Presidents j)

add j to queArray

increment bottom

increment nItems2

//end push

Pop()

Create temp and set to queArrays’ removed item from bottom

Decrement bottom

Decrement nItems2

Return temp

//end Pop

Display()

For( i = 0, if i is less than or equal to bottom, increment i)

Create temp and set to queArrays’ (i) that it “got”

//end Display

peekFront()

Create temp and set to queArrays’ (bottom) that it “got”

Return temp

//end peekFront

isEmpty()

if nItems2 equals 0, return True

//end isEmpty

isFull()

if nItems2 equals size of queArray, return True

//end isFull

//end StackP Class

**(Presidents)**

Declare number and term as integer

Declare name, code, party and state as string

<Constructor> Presidents(int number, String name, String code, int term, String party, String state)

Set code to this.code

Set term to this.term

Set party to this.party

Set state to this.state

//end constructor

getNumber

return number

setNumber(int number)

Set number to this.number

getName

return name

setName(String name)

Set name to this.name

getCode

return code

setCode(String code)

Set code to this.code

getTerm

return term

setTerm(int term)

set term to this.term

getParty

return party

setParty(String party)

set party to this.party

getState

return state

setState(String state)

set state to this.state

//End getters and setters

toString()

return header for arrays

//end toString

compareTo(Presidents o)

set this.name.compareTo to last

return last

//end compareTo

//end Presidents Class