**Public class PrezCollection{**

**PrezCollection()**

Initialize new Presidents Array of size 44 called myPresidents

Initialize int pElem to 0

Initialize new Doubly Linked List called prezList

Initialize 3 new update stacks called addStack, delStack, and invalidStack

//end constructor

**insert(Presidents president)**

Call nameSplitter(president) to get first and last name

Set next element in myPresidents array to president passed as parameter

//end insert

**display()**

For (each president in myPresidents array)

Print president information to formatted string

//end display()

**nameSplitter(Presidents president)**

Initialize new String array of size 3 called nameArray

Initialize String called name to president’s name

Split name at white spaces and assign each substring to an index of nameArray

If (nameArray is length 2)

Set president’s first name to index 0 of nameArray

Set president’s last name to index 1 of nameArray

//end if

If (nameArray is length 3)

Set president’s first name to index 0 of nameArray

Set president’s middle name to index 1 of nameArray

Set president’s last name to index 2 of nameArray

//end if

//end nameSplitter()

**selectionSort()**

declare out, counter for outer loop

declare in, counter for inner loop

declare min, the smallest value in queue

for (out = 0, while out less than pElem-1, increment out)

set min to out

for(in = out+1, while in less than pElem, increment in)

if (President’s last name at position in comes before last name of president at position out)

set min to in

//end if

//end for

swap(out, min)

//end for

//end selectionSort()

**swap(int one, int two)**

declare Presidents object named temp and set to position one in myPresidents

set president object in position one to position two

set president object in position two to temp

//end swap()

**populateList()**

For (each president in myPresidents array)

If (president’s party is republican)

Insert president into doubly linked list prezList

//end if

//end for

//end populateList()

**displayList()**

display doubly linked list prezlist forwards

display doubly linked list back wards

//end displayList()

**insertUpdate(Presidents president)**

call nameSplitter(president) to get first and last name

if (president’s code is add)

push onto addStack

//end if

if (president’s code is delete)

Push onto delStack

//end if

if (president’s code is something else)

push onto invalidStack

//end if

//end getUpdate()

**displayStack()**

display header

display addStack

display header

display delStack

display header

display invalidStack

//end displayStack

**updateList()**

call getUpdateHeader()

for (int i = 0, while i is less than the size of the addStack, increment i)

insert president into prezList

//end for

for(int i = 0, while i is less than the size of the delStack, increment i)

delete president from prezList

//end for

for(int i = 0, while i is less than the size of the invalidStack, increment i)

print formatted invalid transaction message

//end for

//end updateList

**getUpdateHeader()**

print report title

print formatted header

//end getUpdateHeader

//end PrezCollection class