Danny Tan

CS2134

Dt1462

Hw#7

1b)

~List() {

loopPointer = headPointer

while loopPointer is not at end

tempPointer = loopPointer

delete tempPointer

increment loopPointer to next pointer in list

end while

headPointer = nullptr

}

headPtr

loopPtr

nullptr

b

a

headPtr loopPtr

nullptr

b

a

headPtr loopPtr

nullptr

b

headPtr

nullPtr

1c)

front () {

return the data of the Node after head

}

headPtr front

nullptr

b

a

Returns the front so link is not changed

1d)

merge ( List 2) {

mergeList = head of first List (doesn’t matter which head we do)

firstLoop = pointer after head

secondLoop = pointer after List 2 head

while (firstLoop is not at end and secondLoop is not at end)

if firstLoop’s data is less than secondLoop’s data

increment first Loop and mergeList

(next node on MergeList is in 1st list so we don’t need to break link)

else

set link to second loop

increment second loop and mergeList

(next node on MergeList is in 2nd list so we need to break link)

end while

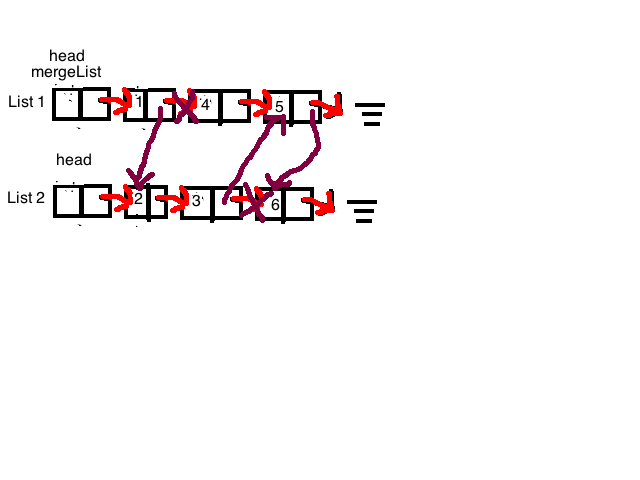
while secondLoop is not at end (if List 2 is bigger than first List)

add whatever is left in List 2 to the mergeList

end while

List 2 headptr = nullptr

}



1e)

remove\_adjacent\_dupe() {

currentPointer = headPointer

while the next pointer is not null pointer

if this data == next data

temp = next pointer

currentPointer = pointer after temp

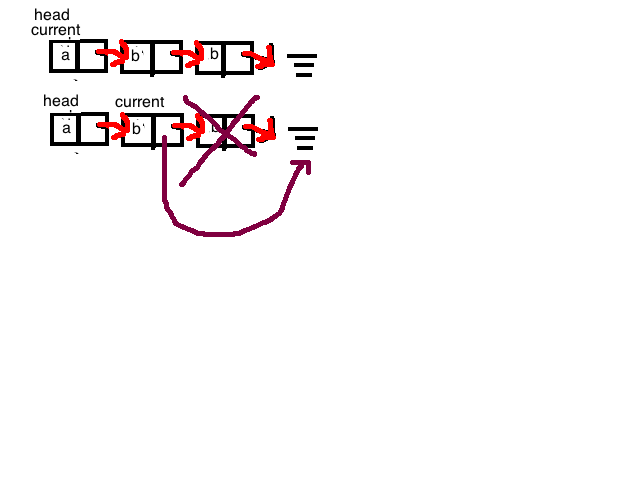
delete temp

else

increment the currentPointer to next pointer

end if

end while

}

1f)

remove\_if (predicate) {

currentPointer = headPointer

while the next pointer is not null pointer

if predicate of the data of the next pointer is true

then erase\_after this pointer

else

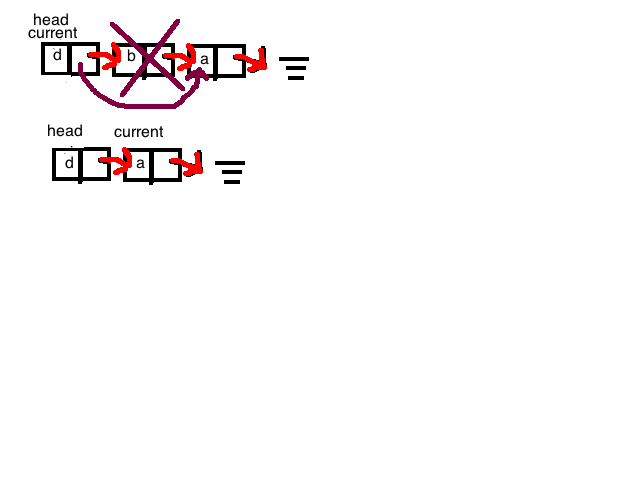
increment the current pointer to next power

end if

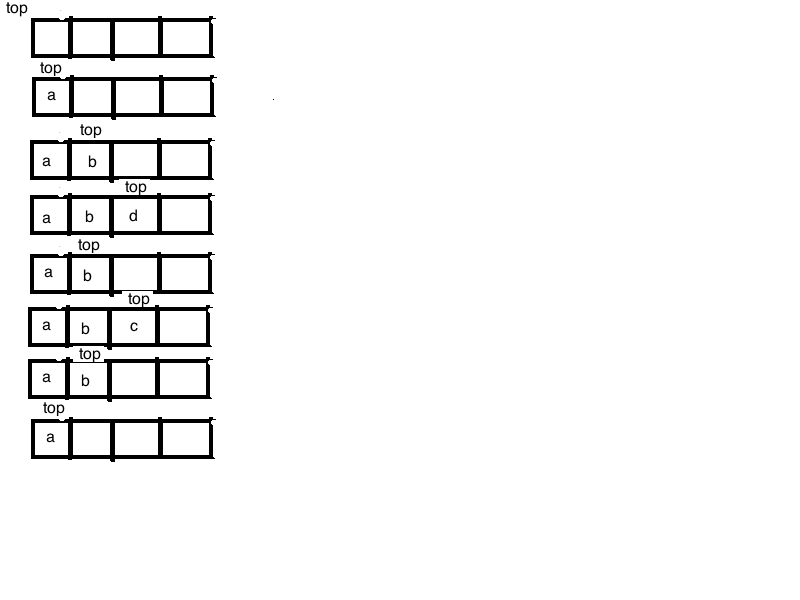
end while

}

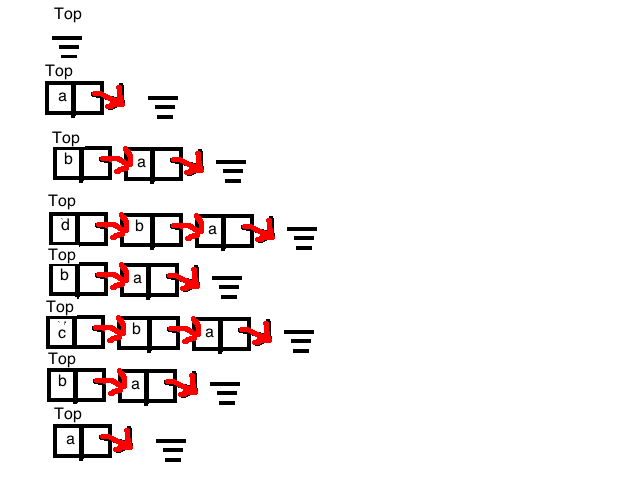
pred(b) is true

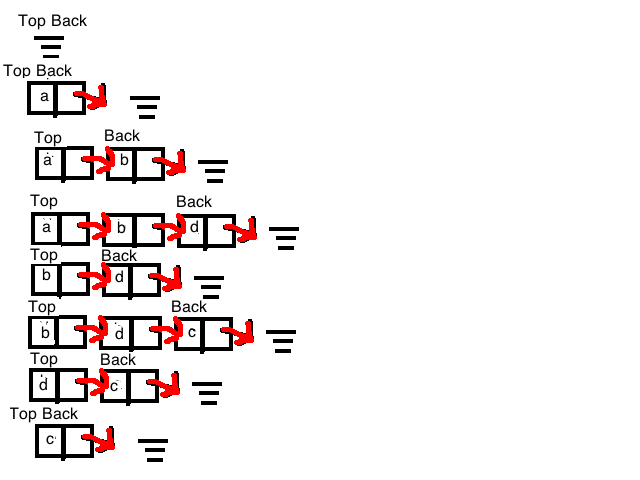


2)



3)



4)

5)