Jake Goldstein

HW 07 Written

1. List Class
   1. Destructor

Header 🡪 |\_\_\_| 🡪 A→B→C→

Header 🡪 |\_\_\_| 🡪 A→B→

Header 🡪 |\_\_\_| 🡪 A→

Header 🡪 |\_\_\_| 🡪

Pseudo Code:

Get the next element in the List

While there are more elements to delete

Delete

Move to the next element

* 1. Front

Header 🡪 |\_\_\_| 🡪 A→B→C→

(No change)

Pseudo Code:

Get the data of the element after the header

* 1. Merge

Header 🡪 |\_\_\_|🡪 1→3→5→ Header 🡪 |\_\_\_| 🡪 2→4→6→ Header→ |\_\_\_|→

Header 🡪 |\_\_\_|→3→5→ Header 🡪 |\_\_\_| 🡪 2→4→6→ Header→ |\_\_\_|→1→

Header 🡪 |\_\_\_|→3→5→ Header 🡪 |\_\_\_| →4→6→ Header→ |\_\_\_|→1→2→

Header 🡪 |\_\_\_|→5→ Header 🡪 |\_\_\_| →4→6→ Header→ |\_\_\_|→1→2→3→

Header 🡪 |\_\_\_|→5→ Header 🡪 |\_\_\_| →6→ Header→ |\_\_\_|→1→2→3→4→

Header 🡪 |\_\_\_|→ Header 🡪 |\_\_\_| →6→ Header→ |\_\_\_|→1→2→3→4→5→

Header 🡪 |\_\_\_|→ Header 🡪 |\_\_\_| → Header→ |\_\_\_|→1→2→3→4→5→6→

Header 🡪 |\_\_\_|→1→2→3→4→5→6→ Header 🡪 |\_\_\_| → Header→ |\_\_\_|→

Pseudo Code:

Create a temp List

While there are still elements in both lists that are being merged

The smaller element gets put into the temp, and erased from their List

Once one of the Lists is empty

Add all the leftover elements to the temp and delete them from their List

Swap the temp header with header

* 1. Remove Adjacent Duplicates

Header 🡪 |\_\_\_| 🡪 X→Y→X→X→Z→

Header 🡪 |\_\_\_| 🡪 X→Y→X→X→Z→

Header 🡪 |\_\_\_| 🡪 X→Y→X→X→Z→

Header 🡪 |\_\_\_| 🡪 X→Y→X→Z→

Header 🡪 |\_\_\_|→X→Y→X→Z→

Pseudo Code:

Make two temps to the first two elements

While there are still elements in the list

If the two temps are the same, delete one

Move to the next element

* 1. Remove If

(In this case Pred is x>5)

Header 🡪 |\_\_\_| 🡪 7→4→2→11→0→4→

Header 🡪 |\_\_\_| →4→2→11→0→4→

Header 🡪 |\_\_\_| →4→2→11→0→4→

Header 🡪 |\_\_\_| →4→2→0→4→

Header 🡪 |\_\_\_| →4→2→0→4→

Header 🡪 |\_\_\_| →4→2→0→4→

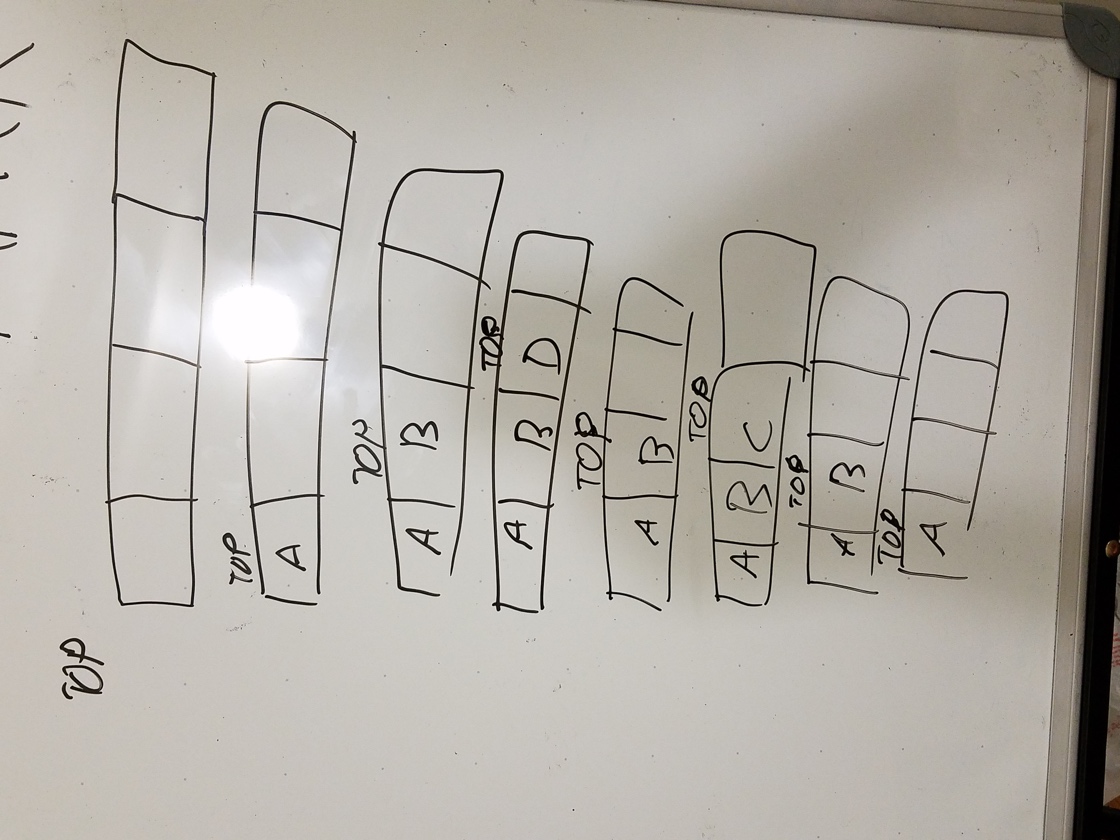
Pseudo Code:

While there are still elements in the list

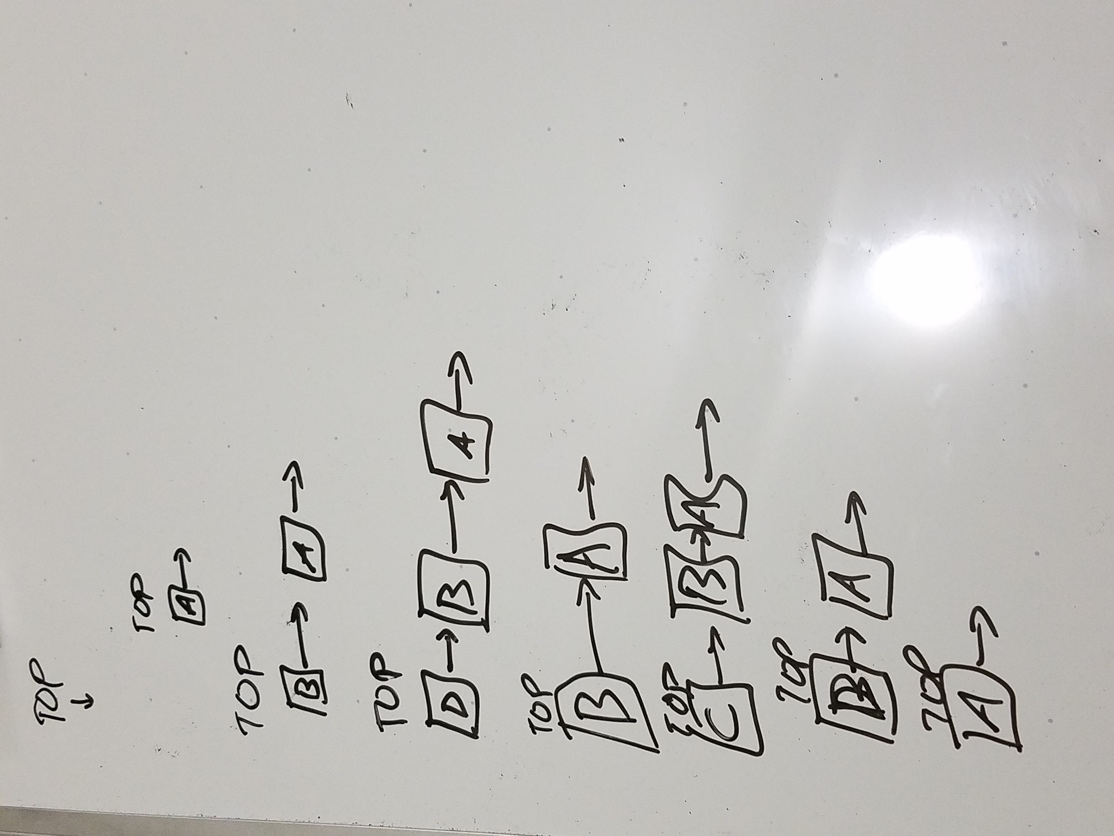
If the element passes the Pred test delete it

Move to the next element

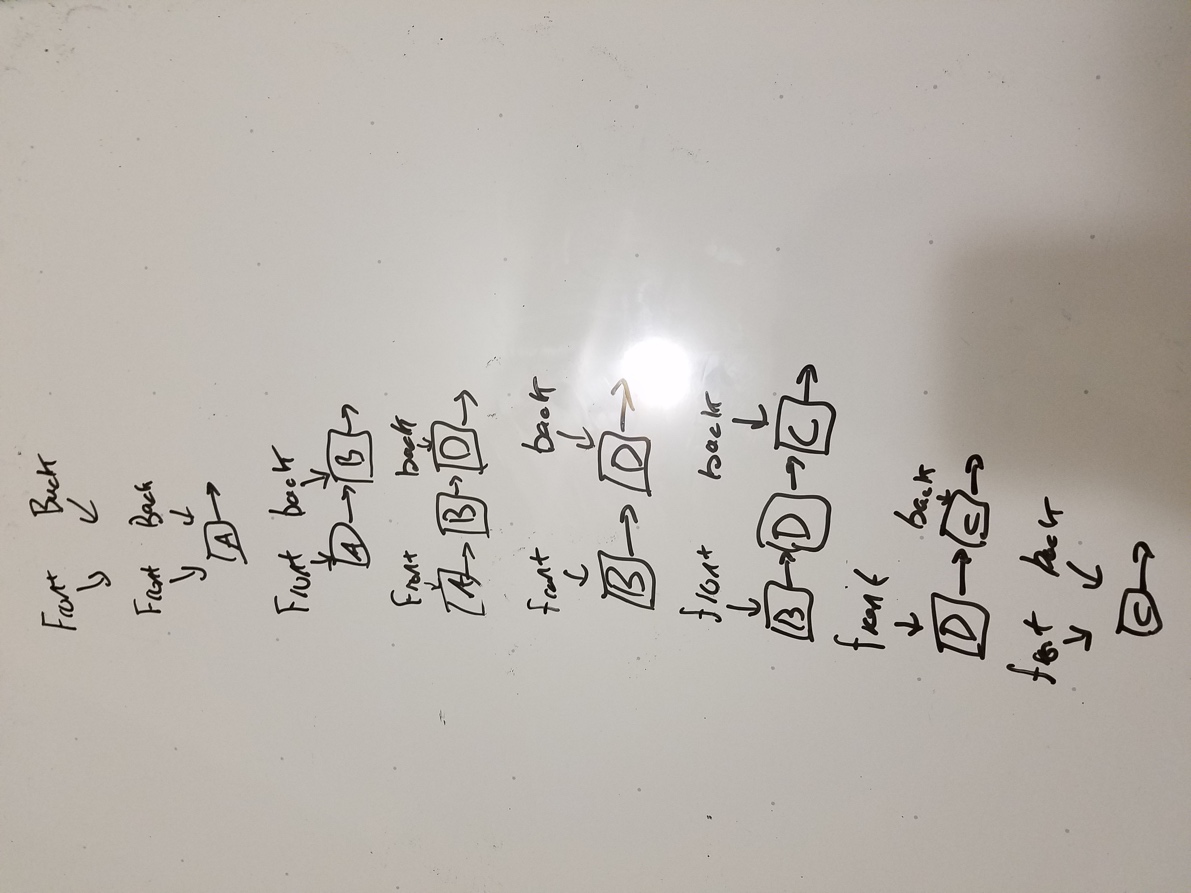
1. static array implementation of the ADT stack with array size MAX = 4



1. linked list implementation of the ADT stack



1. linked list implementation of the ADT queue



1. dynamic array implementation of the ADT queue where it initial size of the array is 4,

