Dynamic Array Deque Implementation
Let the partially filled block “float”

- One solution: Rather than always use index zero as our starting point, allow the starting index to “float”

- Maintain two integer values:
  - Starting or beginning index \( \text{beg} \)
  - Count of elements in the collection \( \text{size} \)

- Still need to reallocate when size equal to capacity
Dynamic Array Deque

- First filled element no longer always at index 0
- Filled elements may wrap around back to the front end of array
- Called **ArrDeque**
struct ArrDeque {
    TYPE *data;    /* Pointer to data array. */
    int size;      /* Number of elements in collection. */
    int beg;       /* Index of first element. */
    int cap;       /* Capacity of array. */
};

void initArrDeque(struct ArrDeque *d, int cap) {
    d->data = malloc(cap * sizeof(TYPE));
    assert(d->data != 0);
    d->size = d->beg = 0;
    d->cap = cap;
}
Adding to back is easy, just adjust count of the number of elements

- Still need to reorganize if adding and size = capacity

**Add**

**Remove**
Changes to front are easy, just adjust count and starting location

**Add**

**Remove**
Problem: *Elements can wrap around from beg to end*
• Calculate offset: add logical (element) index to start \((\text{beg})\)
  
  \[
  \text{offset} = \text{beg} + \text{logicalIndex}; \quad /* \text{logIndex} = 3, \text{offset} = 9 */
  \]

• If larger than or eq to capacity, subtract capacity
  
  \[
  \text{if (offset} \geq \text{cap) }
  \]
  
  \[
  \text{absoluteIndex} = \text{offset} - \text{cap};
  \]

• If smaller than zero, add cap
  
  \[
  \text{if (offset} < 0)
  \]
  
  \[
  \text{absoluteIndex} = \text{offset} + \text{cap};
  \]

• That way sizes are always in range

• Or..combine into single statement with mod:

  \[
  /* \text{Convert logical index to absolute index.} */
  \]
  
  \[
  \text{absIdx} = (\text{logicalIdx} + \text{beg}) \mod \text{cap};
  \]
Can we simply copy the elements to a larger array?

Have to be careful because the wrapping is dependent on the ‘capacity’
## Operations Analysis

<table>
<thead>
<tr>
<th>Operation</th>
<th>Best</th>
<th>Worst</th>
<th>Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddBack</td>
<td>1</td>
<td>n</td>
<td>1+</td>
</tr>
<tr>
<td>RemoveBack</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>AddFront</td>
<td>1</td>
<td>n</td>
<td>1+</td>
</tr>
<tr>
<td>RemoveFront</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Your Turn...

• Complete Worksheet 20