# **Representations and Abstractions**

1. (1 point) Write a CircArrayPipe representation for the Pipe abstraction = [P, Q, R]:4 in which last > first and first ≠ 0.

|  |
| --- |
| contents = [null, P, Q, R]  first = 1  last = 3  length = 3  capacity = 4 |

1. (1 point) Write a CircArrayPipe representation for the Pipe abstraction = [P, Q, R]:4 in which last < first and none of the array elements are null values.

|  |
| --- |
| contents = [R, Z, P, Q]  first = 2  last = 0  length = 3  capacity = 4 |

1. (1 point) Write the Pipe abstraction corresponding to the CircArrayPipe representation: contents = [P, Q, R, S, T] and first =3 and length = 4.

|  |
| --- |
| pipe = [S, T, P, Q]:5 |

1. (2 points) Give both the representation and abstraction for the CircArrayPipe built using the following code sequence (assume an initial array contains all null values)  
     
   Pipe<String> pipe = new CircArrayPipe<>(4);  
   pipe.append("W");  
   pipe.append("X");  
   String s1 = pipe.removeFirst();  
   pipe.prepend("Y");  
   String s2 = pipe.removeLast();  
   pipe.prepend("Z");  
   pipe.prepend(s2);

|  |
| --- |
| contents = [Y, null, X, Z]  first = X  last = Y  length = 3  capacity = 4 |

|  |
| --- |
| pipe = [X, Z, Y]:4 |

1. (2 points) We implemented circular array pipe with fields first, last, and length. How would you implement a length method using only fields first, last, capacity, and math operations (including %)? Do not use an if statement. Your method should have a single line.

|  |
| --- |
| public int length() {  return last == -1 ? 0 : (last - first + capacity()) % capacity() + 1;  } |

Figure 1 (before)

Diagram

Description automatically generated

Figure 2 (after)

A picture containing clock, object, drawing

Description automatically generated

1. (2 points) Consider the diagram in Figure 1 that shows a doubly linked list along with a temporary node that holds the element W. The dashed arrows are previous links, and the solid arrows are next links. Arrows with a circle at the end point to null. We want to end up with a doubly linked list like the one in Figure 2. What are the next two steps in order to achieve that? Choose from one of the following statements for each step, where [var1] and [var2] are either temp, first, or last, and [previous|next] is either previous or next.
   1. Relocate [var1] to [var2]'s node
   2. Redirect [var1]'s [previous|next] link to [var2]'s node
   3. Make [var] follow its [previous|next] link

|  |
| --- |
| step 1: Redirect ***first***'s ***previous*** link to ***temp***'s node  step 2: Make ***first*** follow its ***previous*** link |

1. (1 point) What is a one-line command (in Java code) for "Make *first* follow its previous link"?

|  |
| --- |
| *first* *=* *first.previous;* |