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

2. (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

3. (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

4. (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

5. (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)

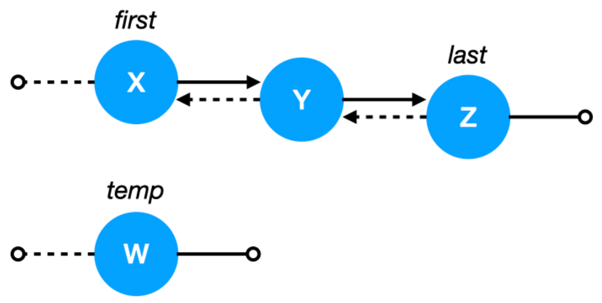
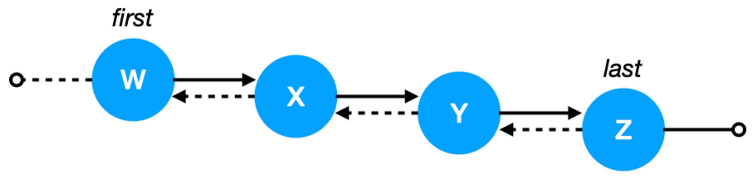


Figure 2 (after)



6. (2 points) The diagram in Figure 1 shows a doubly linked list along with a temp node holding element W. Dashed arrows are previous links, and solid arrows are next links. Arrows with a circle at the end point to null. We want to end up with a list like the one in Figure 2. What are the three steps to achieve that? Choose from one of the following statements for each step. [var1] and [var2] are temp, first, or last; [previous|next] means previous or next.

   a. Relocate [var1] to [var2]'s node
   b. Redirect [var1]'s [previous|next] link to [var2]'s node
   c. Make [var] follow its [previous|next] link

```
step 1: Redirect first's previous link to temp's node
step 2: Redirect temp's next link to first's node
step 3: Relocate first to temp's node
```

7. (1 point) What is a one-line command (in Java code) for "Make *first* follow its previous link"?

```
first = first.previous;
```