5. Hoffery (8 Points). Suppose we have the following generic HOF interface:

```
public interface UnaryFunction<T> {
    public T apply(T x);
}
```

A spy is a `UnaryFunction` that wraps around another `UnaryFunction`, `f`. For any input `x`, it returns the result `f` would return, but it remembers the arguments on which it has been called and can print them out on command. For example, if `SquareFunction` represents a function that squares integers:

```
UnaryFunction<Integer> sq = new SquareFunction();
System.out.println(sq.apply(4)); // prints "16", not including quotes

Spy<Integer> spy = new Spy<>(sq);
System.out.println(spy.apply(5)); // prints "25", not including quotes
System.out.println(spy.apply(2)); // prints "4"
System.out.println(spy.apply(3)); // prints "9"
spy.printArgumentHistory(); // prints "5 2 3 "         non-destructive!
spy.printArgumentHistory(); // prints "5 2 3 "    and not including quotes
```

Complete the Spy class below. You may assume you have access to a working `LinkedList`, if necessary. **Only write one statement per line.**

```
interface LinkedList<E>
void addFirst(E x);
void addLast(E x);
boolean isEmpty();
int size();
E get(int index);
E removeFirst();
E removeLast();
```

```
public class Spy<T> _____________________ {

   ____________________________
   ____________________________
   public Spy(_____________________________) {

     ____________________________
     ____________________________

  }

 @Override
 public T apply(___) {
     ____________________________________________
     ____________________________________________
     ____________________________________________
  }

  public void printArgumentHistory() {
     ____________________________________________
     ____________________________________________
     ____________________________________________
  }
}
```