R-6.1 Suppose an initially empty stack *S* has performed a total of 25 push operations, 12 top operations, and 10 pop operations, 3 of which returned null to indicate an empty stack. What is the current size of *S*?

R-6.2 Had the stack of the previous problem been an instance of the ArrayStack class, from Code Fragment 6.2, what would be the final value of the instance variable t?

R-6.3 What values are returned during the following series of stack operations, if executed upon an initially empty stack? push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop().

R-6.4 Implement a method with signature transfer(*S*, *T*) that transfers all elements from stack *S* onto stack *T*, so that the element that starts at the top of *S* is the first to be inserted onto *T*, and the element at the bottom of *S* ends up at the top of *T*.

R-6.5 Give a recursive method for removing all the elements from a stack.

Answers

R-6.1 The current size of *S* is 15.

R-6.2 The final value of the instance variable t would be 15.

R-6.3 The returned values are: 3, 8, 2, 1, 6, 7, 4, 9.

R-6.4 The implementation would be:

```
public void transfer(ArrayStack s, ArrayStack t) {

    while (!t.isEmpty()) {
        t.push(s.pop());
    }
}
```

R-6.5 The method would be:

```
public void recursiveRemove() {
    if (!isEmpty()) {
        this.pop();
        recursiveRemove();
    }
}
```

```java
public class ArrayStack<E> implements Stack<E> {
    public static final int CAPACITY = 1000; // default array capacity
    private E[] data; // generic array used for storage
    private int t = -1; // index of the top element in stack

    public ArrayStack() {
        this(CAPACITY);
    } // constructs stack with default capacity

    public ArrayStack(int capacity) { // constructs stack with given capacity
        data = (E[]) new Object[capacity]; // safe cast; compiler may give
warning
    }

    public int size() {
        return (t + 1);
    }

    public boolean isEmpty() {
        return (t == -1);
    }

    public void push(E e) throws IllegalStateException {
        if (size() == data.length)
            throw new IllegalStateException("Stack is full");
        data[++t] = e; // increment t before storing new item
    }

    public E top() {
        if (isEmpty())
            return null;
        return data[t];
    }

    public E pop() {
        if (isEmpty())
            return null;
        E answer = data[t];
        data[t] = null; // dereference to help garbage collection
        t--;
        return answer;
    }
}
```

Code Fragment 6.2: Array-based implementation of the Stack interface[1].

[1] From “Data Structures and Algorithms in Java™” book, page 230.