Error Log for Task 3: StackList

Error 1: Missing Check for Empty Stack in `pop()` and `peek()`

- **Description**:
  The `pop()` and `peek()` methods did not handle the case where the stack was empty. Calling these methods on an empty stack resulted in undefined behavior or crashes.

Fix:
I added a check in both methods to verify if the stack is empty using the `isEmpty()` method. If the stack is empty, an `IllegalStateException` is thrown to prevent unsafe operations.

Code Fix Example:

```
public T pop() {
    if (isEmpty()) {
        throw new IllegalStateException("Cannot pop from an empty stack.");
    }
    T data = top.data;
    top = top.next; // Update top pointer
    size--; // Update size
    return data;
}


public T peek() {
    if (isEmpty()) {
        throw new IllegalStateException("Cannot peek at an empty stack.");
    }
    return top.data; // Return the top element without removing it
```

```
}
```

-

Error 2: Incorrect Handling of the `top` Pointer During `push()` Operation

- **Description**:
  When pushing an element onto the stack, the `top` pointer was not correctly updated to link the new node to the existing stack. This caused the stack to lose references to newly added elements and broke its structure.

  Fix:
  I updated the `push()` method to properly link the new node's `next` pointer to the current `top` and then set the `top` pointer to the new node. This ensured that the stack's linked list structure was maintained correctly.

  Code Fix Example:

```
public void push(T data) {
    Node newNode = new Node(data);
    newNode.next = top; // Link new node to the current top
    top = newNode;      // Update top pointer to the new node
    size++;             // Update size
}
```

-

Error 3: Incorrectly Updating the Stack Size

- **Description**:
  The `size` field was not updated during `push()` and `pop()` operations, leading to incorrect results when calling the `size()` method.

  Fix:
  I ensured that the `size` field was incremented after each `push()` operation and

decremented after each `pop()` operation. This ensured that the `size()` method returned the accurate number of elements in the stack.

Code Fix Example:

```
public int size() {
    return size;
}

public void push(T data) {
    Node newNode = new Node(data);
    newNode.next = top;
    top = newNode;
    size++; // Increment size
}

public T pop() {
    if (isEmpty()) {
        throw new IllegalStateException("Cannot pop from an empty stack.");
    }
    T data = top.data;
    top = top.next;
    size--; // Decrement size
    return data;
}
```

-