분반 : 01
학번 : 201802092
이름 : 박민

문제 해결 방법

0) 테스트 케이스

1~4번 까지만 구현하였으며 테스트 케이스와 결과는 다음과 같다. 각 출력문에 어떤 메서드를 의미하는 케이스 인지 주석으로 명기하였다.

```java
/* (4) toString */
System.out.println(list.toString());

/* (1) linkLast */
list.add( index: 8,  element: 'b');
System.out.println(list.toString());

/* (2) node */
System.out.println(list.get(0));

/* (3) length */
System.out.println(list.size());
```

```
[ a b c d d c b a ]
[ a b c d d c b a b ]
a
9
```

1) linkLast

```java
private void linkLast(char element, Node x) {
    if(x.next == null) {
        // 맨 끝이라면 노드를 추가
        Node nexthead = new Node(element,  next: null);
        x.next = nexthead;
    } else {
        // 아니라면 다음 노드를 탐색
        linkLast(element, x.next);
    }
}
```

재귀를 리스트의 가장 끝으로 이동하는 방법으로 사용하였다. 현재 노드의 다음 노드(next)를 탐색하고 만약 null이라면 리스트의 끝이라는 뜻이므로 (base-case) 노드를 하나 추가한다.

2) node

```
private Node node(int index, Node x) {
    return (index == 0) ? x : node( index: index - 1, x.next);
    // offset 만큼 가기 위해 index를 1씩 줄여감
}
```

입력받은 index가 0이 될 때 까지 1씩 줄여간다. 만약 index가 0이 된다면 해당 노드를 리턴한다.

3) length

```
private int length(Node x) {
    return (x.next == null) ? 1 : 1 + length(x.next);
    // 다음이 null일 때 까지 length를 1씩 늘려감
}
```

리스트의 끝에(null) 도달할 때까지 재귀 호출한다. 이 때 1씩 더해주면 맨 마지막에 call stack이 정리되며 길이가 리턴된다.

4) toString

```
private String toString(Node x) {
    return (x == null) ? "" : x.item + " " + toString(x.next); // null 일 때 까지 출력
}
```

리스트의 끝이 나올 때까지 재귀 호출을 이용하여 문자열을 합쳐준다.