Linked List

Đoạn code java phía dưới là thiết kế chi tiết của lớp LinkedList, và phương thức main để kiểm tra. Các em hãy hoàn thiện lớp này với gợi ý sau:

LinkedList<T>: Là kiểu dữ liệu Generic, cho phép lưu trữ tập hợp các phần tử kiểu Number. LinkedList<T>, như tên đã thể hiện, lưu trữ dữ liệu ở dạng danh sách liên kết đơn (Tài liệu tham khảo, Chapter 3), hỗ trợ các methods như trong file thiết kế.

Input

+ Dòng đầu tiên gồm hai số n (0 <= n <= 2*10^5), m (0 <= m <= 100): số lượng phần tử ban đầu của danh sách, và số lượng câu lệnh (command)

+ Dòng tiếp theo chứa n phần tử ban đầu của danh sách

+ m dòng tiếp theo chứa các câu lệnh theo dạng: "command [parameters]". Command là tên method, parameters là tham số tương ứng với file thiết kế

Output

+ Với các command có trả ra giá trị (sum, average, getAt, size, firstIndexOf, lastIndexOf), xuất kết quả vào standard output

Example

Input:
10 6
2 4 -4 -3 -2 -3 3 1 -5 -3
getAt 0
getAt 8
firstIndexOf 1
lastIndexOf 1
sum
average

Output:
2
-5
7
7
-10.0
-1.0

Program

```
import java.util.Scanner;

class LinkedList<T extends Number> {

  static private class LinkedNode<U extends Number> {
    U number;
    LinkedNode<U> next;
```

```
  public LinkedNode(U number) {
    this.number = number;
  }
}

LinkedNode<T> head = null;

private int compare(T n1, T n2) {
  long l1 = n1.longValue();
  long l2 = n2.longValue();
  if (l1 != l2) {
    return (l1 < l2 ? -1 : 1);
  }
  return Double.compare(n1.doubleValue(), n2.doubleValue());
}

public int size() {
  // Your code here
  return 0;
}

public void add(T number) {
  LinkedNode<T> newNode = new LinkedNode<T>(number);
  // Your code here
}

/**
 * @return -1 if number is not in list
 */
public int firstIndexOf(T number) {
  // Your code here
  return -1;
}

/**
 * @return -1 if number is not in list
 */
public int lastIndexOf(T number) {
  // Your code here
  return -1;
}


/**
 * Remove first occurence of number
 */
public void removeFirst(T number) {
  // Your code here
}

public void removeAt(int index) {
  // Your code here
}

public void insertAt(int index, T number) {
  // Your code here
}

/**
```

```
     * @return null if index is out of range
     */
    public T getAt(int index) {
        // Your code here
        return null;
    }

    public double sum() {
        // Your code here
        return 0;
    }

    public double average() {
        // Your code here
        return 0;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        LinkedList<Integer> linkedList = new LinkedList<Integer>();
        // You code here
    }
}
```

Gợi ý testcases:

Có 19 testcases, 50% là testcases đơn giản chỉ kiểm tra các tác vụ riêng lẻ. Các em có thể sumit từng phần để kiểm tra xem đúng được những tác vụ nào:

1. Testcase 1, 18: Kiểm tra tác vụ getAt
2. Testcase 2: Kiểm tra tác vụ firstIndexOf
3. Testcase 3: Kiểm tra tác vụ lastIndexOf
4. Testcase 4: Kiểm tra tác vụ sum, average
5. Testcase 5: Kiểm tra tác vụ getAt, firstIndexOf, lastIndexOf, sum, average, size (CHECKS)
6. Testcase 6: Kiểm tra tác vụ removeAt và CHECKS
7. Testcase 7, 19: Kiểm tra tác vụ insertAt và CHECKS
8. Testcase 8: Kiểm tra tác vụ removeFirst và CHECKS
9. Testcase 9-11: Kết hợp 2/3 tác vụ trên và CHECKS
10. Testcase 12-14: Kiểm tra với danh sách ít nhất 1000 phần tử
11. Testcase 15-17: Kiểm tra với danh sách ít nhất 100000 phần tử! Lưu ý vấn đề thời gian thực thi