Linked List Unit 1

Q1 Write a code to display contents of a linked list in java accept data in main
input
10 20 30 40 50 -1
Output
10 20 30 40 50
-1 indicates end of linked list

Input
5 4 3 2 1 -1
Output
5 4 3 2 1

Input
1 2 3 4 5 -1
Output
1 2 3 4 5

```
import java.util.Scanner;

class Node {
   int data;
   Node next;

   Node(int data) {
      this.data = data;
      this.next = null;
   }
}

class LinkedList {
   private Node head;

   // Method to insert data into the linked list
   void insert(int data) {
      Node newNode = new Node(data);
      if (head == null) {
         head = newNode;
      } else {
         Node temp = head;
         while (temp.next != null) {
            temp = temp.next;
         }
         temp.next = newNode;
      }
```

```
    }

    // Method to display the linked list
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        LinkedList list = new LinkedList();

        while (true) {
            int value = s.nextInt();
            if (value == -1) {
                break;
            }
            list.insert(value);
        }
        s.close();

        // Display the linked list
        list.display();
    }
}
```

Q2 Given a linked list, find and return the length of input LL.
Sample Input :
3 4 5 2 6 1 9 -1
Sample Output :
7

Input
10 20 30 40 50 -1
Output
5

Input
5 10 15 20 25 30 -1
Output
6

```
import java.util.Scanner;

class Node {
    int data;
    Node next;


    Node(int data) {
        this.data = data;
        this.next = null;
    }
}


class LinkedList {
    private Node head;

    // Method to insert data into the linked list

    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }


    // Method to calculate the length of the linked list
    int getLength() {
        int count = 0;
        Node temp = head;
        while (temp != null) {
            count++;
            temp = temp.next;
        }
        return count;
    }
}


public class LinkedListLength {
```

```
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        LinkedList list = new LinkedList();

        // Read input until -1 is encountered
        while (true) {
            int value = s.nextInt();
            if (value == -1) {
                break;
            }
            list.insert(value);
        }
        s.close();

        // Get and print the length of the linked list
        System.out.println(list.getLength());
    }
}
```

Q3 Print ith node

For a given a singly linked list of integers and a position 'i', print the node data at the 'i-th' position.
Assume that the Indexing for the singly linked list always starts from 0.

If the given position 'i', is greater than the length of the given singly linked list, then don't print anything.

Sample Input
3 4 5 2 6 1 9 -1
3
Sample Output
2

Input
10 20 30 40 50 -1
2
Output
30

Input
10 20 30 40 50 60 -1
12
Output

```
import java.util.Scanner;

class Node {
    int data;
    Node next;
```

```
    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

class LinkedList {
    private Node head;

    // Method to insert data into the linked list
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }

    // Method to print the i-th node data
    void printIthNode(int i) {
        Node temp = head;
        int count = 0;

        while (temp != null) {
            if (count == i) {
                System.out.println(temp.data);
                return;
            }
            temp = temp.next;
            count++;
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        LinkedList list = new LinkedList();

        // Read linked list input until -1 is encountered
        while (true) {
            int value = s.nextInt();
```

```
            if (value == -1) {
                break;
            }
            list.insert(value);
        }

        // Read the index i
        int i = s.nextInt();
        s.close();

        // Print the i-th node
        list.printIthNode(i);
    }
}
```